# Alibaba Cloud Container Service



Document Version20190701

# 目次

L
2
3
3
7
3
3
6
7
1
3
8
4
2
2
3

# 1 Swarm **クラスター**

# 2 概要

Alibaba Cloud のパワフルなコンピューティング機能をベースにしたディープラーニングソ リューションは、簡単でオープンなエンドツーエンドディープラーニングサービスプラット フォームを提供しています。このソリューションにより、データ科学者およびアルゴリズムエン ジニアは、Alibaba Cloud リソース (ECS (Elastic Compute Service) インスタンス、GPU イ ンスタンス、Alibaba Cloud HPC、OSS (Object Storage Service)、Elastic MapReduce および Server Load Balancer を含む)を使用して、データ準備、モデル開発、モデルトレーニ ング、評価、予測、およびその他のタスクを迅速に実行できます。このソリューションにより、 ディープラーニング機能の API へ転送が簡単になり、業務アプリケーションとの統合が促進され ます。

ディープラーニングソリューションには、以下の特長があります。

- シンプル: ディープラーニングプラットフォームの構築および管理のためのしきい値を下げます。
- ・効率的: CPU や GPU などの異種のコンピューティングリソースの効率を向上し、統合された ユーザーエクスペリエンスが提供されます。
- オープン: TensorSlow、Keras および MXNet などの複数のメインストリームディープラー
   ニングフレームワークをサポートし、カスタマイズ環境がサポートされます。
- フルサイクル: Alibaba Cloud のパワフルなサービスシステムに基づいたエンドツーエンドの ディープラーニングタスクプロセスを構築するためのベストプラクティスが提供されます。
- ・サービス指向: ディープラーニング機能をサービスに変換し、クラウド上のアプリケーション
   と簡単に統合します。

#### はじめに

- 1. 環境を準備します。
  - コンテナークラスターの作成. OSS データボリュームを使用して、 データを保存しま す。OSSFS データボリュームの作成
- 2. Jupyter 環境の作成Jupyter 環境における Git を利用したコード管理をご参照ください。
- 3. スタンドアロンモデルトレーニングまたは分散モデルトレーニングを実行して、モデルをエク スポートします。
- 4. エクスポートされたモデルを利用して、TensorFlow Serving の利用を行います。

# 3環境の準備

### 3.1 データボリュームの作成

OSSFS は FUSE をベースにしたファイルシステムで、Alibaba Cloud により公式に提供され ています (プロジェクトのホームページは、*https://github.com/aliyun/ossfs*をクリックしご参照 ください)。 OSSFS データボリュームは、OSS (Object Storage Service) バケットをデータボ リュームとして パッケージ化できます。

OSSFS のパフォーマンスおよび機能は、ローカルファイルシステムのパフォーマンスおよび機 能とは異なります。ネットワークを通じてデータをクラウドと同期させる必要があるためです。 OSSFS のログなどのファイルを絶えず書き換える必要があるデータベースやアプリケーションな ど、I/O 集約型アプリケーションは実行しないことを推奨します。 OSSFS は、コンテナー間の 設定ファイルの共有や、書き換えを必要としない添付ファイルのアップロードなどのシナリオに 適用できます。

OSSFS は、以下の点でローカルファイルシステムとは異なります。

- ・ランダム書き込みや追加書き込みにより、ファイル全体が上書きされます。
- ・システムがリモートで OSS サーバーへアクセスする必要があるため、リストディレクトリなどのメタデータ操作のパフォーマンスが低下します。
- ファイルやフォルダーの名称変更操作はアトミックではありません。
- ・複数のクライアントが同じ OSS バケットに接続されている場合、各クライアントの動作を自 ら調整します。たとえば、複数のクライアントによる同じファイルへの書き込みを避けます。
   ・ハードリンクはサポートされません。

#### 前提条件

データボリューム機能を有効化するために、クラスターは次の2つの条件を満たす必要がありま す。 ・クラスターエージェントのバージョンが0.6以降であること。

クラスターリストのページでエージェントのバージョンを参照できます。 [詳細] > [エージェ ントのアップグレード] をクリックします。

Upgrade Age	nt - EGS-cluster	$\times$
	<ul> <li>The selected Cluster: EGS-cluster</li> <li>1. The current Agent version: 0.10-10e99cd (latest:0.10-10e99cd)</li> <li>2. The upgrade has no effect on your applications. But during the upgrade, you cannot use the Web interface to manage the cluster, neither can you use Docker clients to connect to the access port of the cluster. The upgrade takes about 2 minutes.</li> <li>3. After clicking "OK", you can close the dialog box and continue to perform other cluster operations.</li> </ul>	
	ОК	Cancel

エージェントのバージョンが 0.6 以前の場合、エージェントをアップグレードします。エー ジェントのアップグレード方法について詳しくは、エージェントのアップグレードをご参照く ださい。

クラスターに acsvolumedriver アプリケーションがデプロイされていること。最新バージョンへのアップグレードを推奨します。

システムサービスのアップグレードにより、acsvolumedriver アプリケーションのデプロイ およびアップグレードができます。 詳しくは、システムサービスのアップグレードをご参照く ださい。

acsvolumedriver がアップグレードまたは再起動された場合、使用中の OSSFS データボ リュームが再起動され、お使いのサービスも再起動されます。

#### 手順

手順 1: OSS バケットの作成

**OSS コンソールにログインして、バケットを作成します。 詳しくは、バケットの作成をご参照く** ださい。

この例では、中国 (深セン) に配置されるバケットが作成されます。

tensorflow-sample	Type Standard Stora	ge Region China South 1 (Shenzhen) Cre	ated At 06/28/20	17, 14:35 Delete Bucket				
Overview   Files Basic Settings Domain Names Image Processing   Basic Data Hotspot Statistics API Statistics Object Access Statistics								
Basic Data	Basic Data							
① Data in the Overview page and Bucket Overview page is	not in real time. It is delayed for two to three hours.							
Storage Used Total Used 🗸	Internet Traffic This Month $$$ Inbound $\lor$	Requests This Month 🛛 🗸 🗸 🗸 РОТ 🗸	Files	File Fragments ⑦				
852.2 кв	0 Byte	0	2	0				
Month-On-Month 0.00% Day-On-Day 0.00%	Internet Traffic Last Month 0Byte	Requests Last Month 0						
Access Domain Name								
	Endpoint (?)	Access Domain Name 💿		HTTPS				
Internet Access ③	OSS-C Halfred Charles Device Com	tensorflow-Lange data and	ecches allywhos.com	Suppor				
ECS Address for Classic Network Access (Intranet) $$	OSS-01-17-01-010-010-010-019-00-00-00-	tensorflow-upper and	enchers-internal allu	Suppor				
ECS Address for VPC Network Access (Intranet) ⑦	OSS-1	tensorflow-	erchen-internal ally	Suppor				

- 手順 2: OSSFS データボリュームの作成
- 1. Container Service コンソールにログインします。
- 2. 左側のナビゲーションウィンドウから [データボリューム] をクリックします。
- 3. [クラスター] ドロップダウンリストから、データボリューム (この例では、"tfoss") を作成す るクラスターを選択します。 右上の [作成] をクリックします。

Container Service	Data Volume Lis	t					Refresh Create
Overview	Help: Ø Data v Cluster: EGS-d	olume guide					3
Services	Node	Volume Name	Driver	Mount Point	Container	Volume Parameters	Action
Clusters				<li>⑦ Could not find an</li>	y record that met the condition.		
Nodes							
Networks							
Data Volumes							
Configurations							

4. [データボリュームの作成] ダイアログボックスが表示されます。 [データボリュームタイプ] と して [OSS] を選択します。データボリュームパラメーターを設定して、[作成] をクリックしま

### す。 Container Service により、クラスターのすべてのノードに同名のデータボリュームが 作成されます。

Create Data Volume	• ×
Type:	OSS      Cloud Disk
Name:	tfoss
Access Key ID:	differently.
Access Key Secret:	-Species provi
Optional Parameters:	🗷 allow_other 🕖 🗉 noxattr 🥥
Other Parameters:	For the formats of other parameters, refer to this document. Example: -o allow_other -o default_permission=666 -onoxattr Note: Only clusters with volume driver version 0.7 or above support these parameters. You can go to the application list, find the acsvolumedriver application, and view the volumedriver service's image version in the service list on the application details page. If the image version is lower than 0.7, please upgrade the volumedriver.
Bucket ID:	Select Bucket
Access Domain Name:	○ Intranet ○ Internet ⑧ VPC ②
File Caching:	Enable
	<b>Create</b> Cancel

- ・名前: データボリューム名はクラスターで一意でなくてはなりません。
- Access Key ID/Access Key シークレット: OSS へのアクセスに必要な Access Key で
   *f*。 Access Key コンソールから取得できます。
- ・バケット ID: 使用する OSS バケットの名称です。 [バケットの選択] をクリックします。表示されたダイアログボックスでバケット (この例では、"tensorflow-sample") を選択して [選択] をクリックします。
- ・アクセスドメイン名: [VPC] を選択します。

ファイルキャッシュ: 複数のマシン上にある同一ファイルの変更を同期させる場合は [無効化] を選択します (たとえば、マシン A 上のファイルを変更して、マシン B で変更した内容を読み込む場合)。

### **注**:

同じフォルダにたくさんのファイルがある場合、ファイルキャッシュの無効化によりフォ ルダの "ls" が特に遅くなります。 前述の要件がない場合、ファイルキャッシュを有効化し て "ls" を高速化します。

#### その後の操作

データボリューム作成後、アプリケーションで利用できます。 アプリケーションでのデータボ リュームの使用方法については、サードパーティのデータボリュームの使用をご参照ください。

### 3.2 コンテナークラスターの作成

ディープラーニングソリューションは、ECS (Elastic Compute Service) インスタンスまたは GPU インスタンスを備えたコンテナークラスターをサポートしています。 このドキュメントで は、例として GPU インスタンスを備えたコンテナークラスターを使用します。

**首**注:

ECS インスタンスを備えたコンテナークラスターの作成方法については、クラスターの作成を ご参照ください。

#### 制限

- ・現在、Container Service は以下のリージョンにおいて GN4 GPU インスタンスを備えたクラスターの作成をサポートしています: 中国 (深セン)、中国 (上海)、中国 (北京) および米国 (シリコンバレー)。
- ・現在、GN4 GPU インスタンスは VPC (Virtual Private Cloud) のみのサポートです。

#### 前提条件

現在、従量課金 GPU Compute タイプ GN4 インスタンスを有効化する必要があります。 有効化 には以下のように、*ECS* チケットを起票し、サポートセンターへお問い合わせください。

「従量課金 GPU Compute タイプ GN4 インスタンスを有効化を希望します。 よろしくお願いし ます。」

#### 手順

1. Container Service コンソールにログインします。

左側のナビゲーションウィンドウから [Swarm] > [クラスター] をクリックします。 右上にある [クラスターの作成] をクリックします。

Container Service	Cluster List			Yo	u can create i	up to 5 cluster	s and can add u	p to 20 nodes in ea	ch cluster.	Refresh	Create Cluster	•
Overview	Help: ${\mathscr S}$ Create cluster ${\mathscr S}$ How to add	d existing ECS insta	nces 🔗 Cross-zone n	ode management 🔗 Log	Service integr	ration 🔗 Con	nect to cluster t	hrough Docker Clie	nt		2	
Applications	Name 🔻											
Services	Cluster Name/ID	Cluster Type	Region	Network Type	Cluster Status	Node Status 🕜	Number of Nodes	Time Created	Docker Version		1	Action
Nodes	test m/HR-56210972340c4067481c28c044239	Alibaba Cloud Cluster	China East 1 (Hangzhou)	VPC vpc- liptions/#failing/spinore	Running	Healthy 🕽	2	2018-02-05 09:44:57	17.06.2-ce	Ma	nage   View Lo D Monitor   1	ogs   elete More •

3. 以下の設定を完了させます。 この例では、"EGS-cluster" という名前を設定したクラスター を中国 (深セン) リージョンに作成します。

* Cluster Name	EGS-cluster							
	The cluster name	should be 1-63 o	haracters long, a	nd can contain n	umbers, Chinese	characters, Englisl	h letters and hyp	hens.
Region :	China North 1	China North 2	China East 1	China East 2	China South 1	Asia Pacific NE 1	US West 1	Asia Pacific SE 1
	(Qingdao)	(Beijing)	(Hangzhou)	(Shanghai)	(Shenzhen)	(Tokyo)	(Silicon Valley)	(Singapore)
	Asia Pacific SE 2	EU Central 1	US East 1		China North 3	Asia Pacific SE 3		
	(Sydney)	(Frankfurt)	(Virginia)	Hong Kong	(Zhangjiakou)	(Kuala Lumpur)		
Zone :	China South 1 Z	one B 🛛 🔻						

・クラスター名: 作成するクラスター名です。1 文字以上 64 文字以内で、数字、漢字、英字 およびハイフン (-) を含むことができます。



クラスター名は、同じアカウント、同じリージョン内で一意である必要があります。

・リージョン: クラスターをデプロイするリージョンを選択します。 [中国 (深セン)] 、[中国 (上海)]、[中国 (北京)] または [米国 (シリコンバレー)] を選択します。

# 🧾 注:

現在、Container Service では以下のリージョンにおいて GN4 GPU インスタンスを持つ クラスターの作成がサポートされます: 中国 (深セン)、中国 (上海)、中国 (北京) および米 国 (シリコンバレー)。

・ゾーン: クラスターのゾーンを選択します。

🗎 注:

サーバーの配分に応じてリージョンおよびゾーンを選択できます。

#### 4. [ネットワークタイプ] として [VPC] を選択して、設定を完了させます。

Network Type :	VPC	
	vpc-wz9fv3jx3wqmy67s 🔻	test -
Initial CIDR Block	172.18.0.0/24	Existing CIDR Block of Container Service $\textcircled{O}$
1	This cannot be the same as th Example: 172.18.0.0/24.	e CIDR Block of a VPC or VSwitch. It cannot be modified once created. Valid range: 172.17.0.0/24–172.31.0.0/24.

VPC により、Alibaba Cloud をベースにした独立したネットワーク環境を構築できます。 独 自の仮想ネットワークの完全な制御が可能です。 これには、フリー IP アドレス範囲、CIDR (Classless Inter-Domain Routing) ブロックの分割、ルートテーブルとゲートウェイの設定 が含まれます。

VPC、VSwitchId およびコンテナーの最初の CIDR ブロック (Docker コンテナーが属する サブネット CIDR ブロック)を 指定します。 IP の管理を容易にするため、それぞれの仮想マ シンコンテナーは複数の CIDR ブロックに属し、 コンテナーサブネット CIDR ブロックは仮 想マシンの CIDR ブロックと競合できません。

ネットワークの競合のような問題を避けるために、お使いの独自の VPC と VSwitchId をコ ンテナークラスターに対して 構築することを推奨します。 5. ノードを追加するかどうかを選択します。



新しいインスタンスを複数持つクラスターを作成することも、ゼロノードクラスターの作成後 に既存のインスタンスを追加することもできます。 既存のインスタンスのクラスターへの追 加方法については、「既存のインスタンスの追加」をご参照ください。

- ・追加する場合
  - a. ノードのオペレーティングシステムを選択します。

Operating Syster Ubuntu 14.04 64bit 🔹 🕐

現在、サポートされるオペレーティングシステムは Ubuntu 16.04 64bit および CentOS 7.4 64bit です。

- b. インスタンスの仕様を設定します。
  - [第3世代] を [インスタンスの世代] として、[GPU Compute タイプ gn4] を [イン スタンスファミリー] として選択します。
  - [32 コア、48 GB (ecs.gn4.8xlarge)] または [56 コア、96 GB (ecs.gn4.14xlarge)]
     を [インスタンスタイプ] として選択します。

📋 注:

GN4 GPU インスタンスの使用が承認されていても、この2つのインスタンスタイ プが見つからないのは、この2つのインスタンスに対して利用できるリソースがな いためです。インスタンスを後で購入するか、翌日に購入することを推奨します。

Instance Generation III Generation III Generation IV
Instance Family: Balanced Type Compact Type Optimized Type Network Network Network Network Concrete Type of the Scharced on t
I/O Optimized instance
Instance Type: 2-core, 4GB ( ecs.n4.lar  More instance type, please contact customer service
Instance Quantity 10set(s) 20set(s) 40set(s) 2 set(s) Each cluster can contain up to 40 ECS instances.
System Disk Type Ultra Cloud Disk SSD Cloud Disk
Data Disk Type: Ultra Cloud Disk SSD Cloud Disk
Attach Data Disk: 🗌 Attach Data Disk
Login: Key Pair Password
<ul> <li>* Logon Passwor</li> <li>The password should be 8-30 characters long and contain three types of characters (uppercase/lowercase letters, numbers and special characters).</li> <li>During cluster creation, we will use this password for node configuration. The password will not be stored.</li> </ul>
* Confirm Passwa

インスタンス数、データディスク容量 (GPU インスタンスはデフォルトで 20 GB のシス テムディスクを持ちます) およびログインパスワードを設定できます。

# 📋 注:

- [データディスクの接続] チェックボックスがオンの場合、データディスクは / var
   / lib / docker ディレクトリに接続され、Docker イメージおよびコンテ
   ナーのストレージとして使用されます。
- パフォーマンスと管理の点から、独立したデータディスクをホストに接続し、 Dockert ボリュームを利用してコンテナーの永続データを管理することを推奨しま す。

・追加しない場合

[既存のインスタンスの追加] をクリックして既存のインスタンスをクラスターに追加、または、 クラスター作成後の クラスターリストページで [既存のインスタンスの追加] をクリックして既存のインスタンスをクラスターに追加できます。

6. パブリック EIP (Elastic IP) を設定するかどうかを選択します。

ネットワークタイプとして VPC を選択した場合、Container Service はデフォルトで各イン スタンスに対して *EIP* を設定します。 インスタンスの設定が必要ない場合、[パブリック EIP を設定しない] チェックボックスをオンにして、 SNAT ゲートウェイを設定します。

EIP : Do not Configure Public EIP

You must configure the SNAT (refer to the following documents) if a public EIP is not configured. Failure in configuring the SNAT will cause the VPC unable to access the public network. This will affect cluster creation and application deployment. Documents for reference: Configuring SNAT for Linux in a VPC environment to use a server proxy with EIP to access the Internet without a public network ECS instance

7. Server Load Balancer インスタンスを作成するかどうかを選択します。

Server Load Balancer Automatically Create Server Load Balancer
A public network Server Load Balancer instance is created by default while a cluster is created. The billing method is Pay-As-You-Go.

デフォルトで、[Server Load Balancer の自動作成] チェックボックスがオンになってい ます。 チェックボックスがオンの場合、クラスター作成後にインターネット Server Load Balancer インスタンスが作成されます。 クラスターのアプリケーションに Server Load Balancer インスタンスを利用してアクセスできます。 これは、従量課金 Server Load Balancer インスタンスです。

8. [クラスターの作成] をクリックします。

#### その後の操作

クラスターリストページで、クラスターの右側にある [ログの表示] をクリックし、クラスターの 作成処理のログを 表示できます。

Cluster List				You can crea	te up to 5 clus	sters and can ac	ld up to 20 nodes in	n each cluster.	Refresh	Create Cluster
Help: 🖉 Create cluster 🔗 How to	add existing ECS insta	nces 🔗 Cross-zone no	ode management 🔗 Log S	ervice integra	tion 🔗 Conr	ect to cluster th	rough Docker Clier	it		
Name 🔻										
Cluster Name/ID	Cluster Type	Region	Network Type	Cluster Status	Node Status 🕜	Number of Nodes	Time Created	Docker Version		Action
EGS-cluster	Alibaba Cloud Cluster	China South 1 (Shenzhen)	VPC vpc- ventholjschwamy 57 minket:	Running	Healthy 🕽	2	2018-01-17 11:17:41	17.06.2-ce	Manage	View Logs Delete Monitor   More -

# 4 モデル開発

### 4.1 Jupyter 環境の作成

#### 前提条件

モデルトレーニングタスクの実行前に、以下の操作を実行したことを確認します。

- 一定数のエラスティックコンピューティングリソース (ECS (Elastic Compute Service) また は EGS) を含むコンテナークラスターを作成します。詳しくは、コンテナークラスターの作 成をご参照ください。
- ・モデルトレーニング用データの保存に OSS (Object Storage Service) を使用するために、同 じアカウントを使用して OSS バケットを作成します。前述のコンテナークラスターにデータ ボリュームを作成し、OSS バケットをローカルディレクトリとして、トレーニングタスクを実 行するコンテナーにマウントします。詳しくは、データボリュームの作成をご参照ください。

規則

アプリケーションのコードでトレーニングデータの読み込み、およびトレーニングデータの出力 を簡単にするため、トレーニングボリューム上のデータは / input ディレクトリに 保存されま す。コードはこのディレクトリからデータを読み込みます。

#### 手順

- 1. Container Service コンソールにログインします。
- 左側のナビゲーションウィンドウから [Swarm] > [イメージとテンプレート] > [ソリューション] をクリックします。

3. [DevBox] で [起動] をクリックします。

Container Service	Machine Learning		
Kubernetes Swarm			
Overview 1	DevBox	Training	P Prediction
Applications	Develop and debug models with Jupyter and	Train models on CPU, GPU with support for	Run prediction on CPU, GPU with support for
Services	Tensorboard. TensorFlow and Keras are supported.	TensorFlow and Keras. Visualize training with TensorBoard.	TensorFlow Serving. Load balancing and scalability are supported by nature.
Clusters	Launch Guide	Launch   Guide   History	Launch   Guide
Nodes Networks	4		
Data Volumes			
Configurations 😐			
Images and Tem	2		
Docker Images			
Orchestration T			
Solutions 3			
Operation Logs			

- 4. Jupyter 環境を作成するための基本情報を設定します。
  - クラスター: 作成したモデルデプロイアプリケーションをデプロイするクラスターを選択します。
     この例では、"EGS-cluster"を選択します。
  - ・アプリケーション名: 作成するアプリケーションの名前です。1 文字以上 64 文字以内で、 数字、英字およびハイフン (-) を含むことができますが、ハイフン (-) で始めることはでき ません。
  - ・フレームワーク: サポートされるフレームワークには、TensorFlow、Keras および Python が含まれます。
  - ・ GPU: 使用する GPU 数です。 このフィールドで "0" を設定すると、GPU は使用されません。
  - ・データソース: トレーニングデータを保存するデータソースを選択します。OSS によりクラスター上の作成されたデータボリュームの選択、または [ローカルディレクトリ] を選択して絶対パスを入力します。[データソースなし] を選択することもできます。
  - · Jupyter パスワード: Jupyter へのログインに使用するパスワードです。
  - ・モニターの有効化: トレーニングステータスのモニタリングのために、TensorBoard を使用するかどうかを選択します。このチェックボックスをオンにした場合、[ログディレクトリ] フィールドにトレーニングログのパスを入力して、入力したパスがトレーニングコード内のログ出力パスと同じであることを確認します。
  - SSH の有効化: SSH によるサービスへのアクセスを許可するかを選択します。このチェックボックスをオンにした場合、SSH パスワードを入力します。

🗎 注:

### SSH によるサービスへのアクセス方法については、SSH を利用した Jupyter サービスへのア クセスをご参照ください。

Cluster	EGS-cluster •
Application Name	mydevbox The name should be 1-64 characters long, and can contain numbers, English letters and hyphens, but cannot start with a hyphen.
Framework	tensorflow:1.1.0
GPUs	0
Data Source	Select Data Source
Jupyter Password	
Log Directory	<ul> <li>Enable Monitor</li> <li>/output/training_logs</li> <li>Please ensure the same log file directory is used in your code.</li> </ul>
SSH Password	✓ Enable SSH
	ОК

- 5. 設定完了後、[OK] をクリックします。
- 6. [アプリケーションリスト] ページで作成したアプリケーションの名前をクリックします。

Cluster: EGS-	-cluster 🔹 🗷 Hide System Applica	Name *	Q <b>X</b>			
Name	Description	Status	Container Status	Time Created 🔺	Time Updated 🔺	Action
mydevbox	eml solution application	Ready	Ready:2 Stop:0	05/19/2018,22:03:07	05/19/2018,22:05:05	Stop   Update   Delete   Redeploy   Events

7. [ルート] タブをクリックします。 "jupyter" と "tensorboard" で始まる 2 つのリンクが表示 されます。

Services	Containers	Logs	Events	Routes	
Route Add	ress				Action
tensorboar	d-2018081922	006.436	044407442	4446512de	scheidel der en hangehou al containen com
jupyter-20	0015118221006	08946	09404910	12Melcib	

8. "jupyter" で始まるリンクをクリックし、Jupyter のパスワードを入力して Jupyter 環境にア クセスします。 9. "tensorboard" で始まるリンクをクリックして、トレーニング結果を参照します。



10.分散ストレージ上のトレーニングデータは、ローカルの /input フォルダに保存されます。 こ のフォルダからデータを読み込むことができます。

# 4.2 Jupyter 環境における Git を利用したコード管理

1. Jupyter のホームページでターミナルを作成します。

💭 Jupyter		Logout
Files Running Clusters		
Select items to perform actions on them.		Upload New - 2
• •		Text File
	Notebook list empty.	Terminal
		Notebooks Python 3

ターミナルで git clone を実行して、アプリケーションのコードをダウンロードします。

```
git clone https ://{ id }:{ password }@ github . com /{ id }/
test . git
Cloning into ' test '...
remote : Counting objects : 3 , done .
```

remote : Total 3 ( delta 0 ), reused 0 ( delta 0 ), pack - reused 0 Unpacking objects : 100 % ( 3 / 3 ), done . Checking connectivi ty ... done .

{ id } および { password } はそれぞれ、GitHubのユーザー名およびパスワードを 示しています。

3. Jupyter のホームページに戻ります。 表示されたアプリケーションのコードを確認

し、Jupyter を使用して対応するコードを開発できます。

4. ターミナルに戻り、Gitを使用してコードを送信できます。

![](_page_18_Figure_7.jpeg)

# 4.3 SSH を利用した Jupyter サービスへのアクセス

Jupyter 環境の作成の実行時に [SSH の有効化] チェックボックスをオンにした場合、 このドキュ メントの手順に従い、SSH を利用して Jupyter サービスにアクセスできます。

この例では、SSH アクセス用のポートマッピングは 192. 32769->22/tcp です。 192. \*\* は ECS (Elastic Compute Service) インスタンスのプライベート IP アドレスです。 Jupyter サービス へのアクセスには、 この IP アドレスは使用できません。 Jupyter サービスにアクセスするため に、ECS インスタンスのパブリック EIP (Elastic IP) を使用して外部からこれらのサービスに SSH アクセスします。

Services	Containers	Log	s Events	Routes				
Name/ID		Status	Health Check	Image	Port	Container IP	Node IP	Action
mydevbox_ju ec39845611	<b>upyter ()</b> .3a4d27	running	Normal	registry-vpc.cn sha256:fdebb9458	192. :32768->8888/tcp 6006/tcp 192. :32769->22/tcp	17	192.	Delete   Stop   Monitor   Logs   Web Terminal
mydevbox_t 11ddefd632	ensorb 053030	running	Normal	registry-vpc.cn sha256:2b4c70877	192. :32770->6006/tcp	17	192	Delete   Stop   Monitor   Logs   Web Terminal

**ECS コンソール** にログインして、ECS インスタンスにバインドされているパブリック EIP を確 認できます。 このページの例では、パブリック EIP は 39. 252 となります。

![](_page_19_Picture_4.jpeg)

さらに、ECS インスタンスのパブリック EIP を使用して外部から Jupyter に SSH アクセスする ために、最初にセキュリティグループルールを設定し、32769 ポートを開きます。

#### セキュリティグループルールの設定

- 1. ECS コンソールにログインします。
- 左側のナビゲーションウィンドウから [インスタンス] をクリックします。 リージョンを選択 します (このページの例では、[中国 (深セン)]となります)。

Elastic Computing Se	Instance List China North 1 (Qingdao) China North 2 (Beijing) China North 2 (Beijing)	rth 3 (Zhangjiakou) China East 1 (Hangzho	u) China East 2 (Shanghai) China So	Hong Kong Asia Pacific NE 1 (Tokyo)		Create Instance
Overview	Asia Hacino Se 1 (Singapore) Asia Hacino Se 2 (Sydney) U	s East 1 (Virginia) US West 1 (Silcon Valley	) Middle East 1 (Dubal) EO Central 1	(Hankruft) 2		
Instances	Instance Name    Enter instance name (fuzzy search)  Sea	rch 📎 Tag		-	Advanced Search Show All Resource	es 🕕 🕺 🗘 ?
<ul> <li>Block Storage</li> </ul>	Instance ID/Name Monitor Zone	IP Address	Status(All) + Network Type(All) +	Configuration	Billing Method(All) 👻	Action
Cloud Disks NAS	□ i+wz95e6pks2//51dwim5k cfb/fb/5277074/9489625          China South 1 Zone B	(Elastic IP Address) (Private IP Address)	Running VPC	CPU: 4 Core(s) Memory: 30 G8 (I/O Optimized) GPU : NVIDIA M40 100 Mbps ( peak value )	Pay-As-You-Go 17-07-03 15:07 created	Manage   Connect Change Configuration   More -
<ul> <li>Snapshots &amp; Images</li> <li>Snapshots</li> </ul>	□ <sup>L-wz9C9D361inh9l8ezpnp</sup> C5703ff57990949349375b	.252(Elastic IP Address) .87(Private IP Address)	Running VPC	CPU: 4 Core(s) Memory: 30 GB (I/O Optimized) GPU : NVIDIA M40 100 Mbps ( peak value )	Pay-As-You-Go 17-06-28 15:46 created	Manage   Connect Change Configuration More -

Container Service アプリケーション mydevbox に対応する ECS インスタンスの右側にある [詳細] をクリックします。 > ドロップダウンリストから [セキュリティグループの設定] を選択します。

 $\sim$ 

4. Container Service クラスターに対応するセキュリティグループの右側にある [ルールの設 定]をクリックします。

Security Group ID/Name	Description	VPC	Acti	ons
sp-topi 2pctruolipmtzvetkio i 1. verzeloc 7 wst_VVC	-	vpc-bpt.wiki.loging.cl.stmlp.i.coli	Configure Rules	ve

5. [セキュリティグループルールの追加] をクリックします。 [セキュリティグループルールの追 加] ダイアログボックスが表示されます。 ルール情報を入力して [OK] をクリックします。

Add	Security Group Ru	les	×
	NIC:	Intranet	
	Rule Direction:	Inbound	
Aut	thorization Policy:	Allow	
	Protocol Type:	Custom TCP	
	* Port Range:	32769/32769	
	Priority:	1	
Au	thorization Type:	Address Field Access	
	* Authorization Object:	0.0.0/0	Tutorial
	Description:		
		It must contain 2-256 characters and it cannot begin with http:// or https://	

![](_page_20_Picture_6.jpeg)

#### SSH を利用した Jupyter サービスへのアクセス

Linux の場合

Linux マシンを使用している場合、以下のコマンドを実行して、SSH により Jupyter サービス にアクセスします。

ssh - p 32769 root @ 39 . 252

32769 は SSH によりアクセスするポート、 39 . 252 は ECS インスタンスにバインドされた パブリック EIP です。

Windows の場合

1. PuTTY を実行して、セッションを設定します。

IP アドレス (ECS インスタンスのパブリック EIP)、SSH によりアクセスするポート (この ページの例では、32769) を設定し、接続タイプとして [SSH] を選択します。 次に、[開く] を クリックします。

🕵 PuTTY Configuration		×				
Category:						
- Session	Basic options for your PuTTY session					
Terminal	Specify the destination you want to connect to					
Keyboard Bell	Host Name (or IP address)Po39.25232	rt 2769				
Window Appearance	Connection type: Raw Telnet Rlogin SSH	🔘 Serial				
Behaviour Translation Selection Colours	Load, save or delete a stored session Saved Sessions					
Connection     Data     Proxy     Telnet     Rlogin     SSH     Serial	Default Settings	Load Save Delete				
	Close window on exit Always Never Only on clean e	exit				
About	Open	Cancel				

2. 表示されたダイアログボックスで Jupyter にログインしてサービスにアクセスします。

ログインアカウント root を入力します。 [SSH の有効化] で設定した [SSH パスワード] を入 力します。 これは、*Jupyter* 環境の作成により設定されたものです。

SSH により Jupyter サービスにアクセスできます。

![](_page_22_Figure_5.jpeg)

# 4.4 ポート転送による Jupyter サービスへのアクセス

ポート転送による Jupyter サービスへのアクセスには以下のようなメリットがあります。 ただし、設定は複雑なものとなります。

- Server Load Balancer インスタンスの購入が必要ないため、コストを抑えることができます。
- ・ 公式 Web サイトからポートを開かず、外部から Jupyter サービスにアクセスできます。

以下の例では、Jupyter サービスはローカルポート 12345 でアクセス要求を転送することにより、SSHにアクセスします。

Services	Containers	Logs	s Events	Routes				
Name/ID		Status	Health Check	Image	Port	Container IP	Node IP	Action
mydevbox_ju ec39845611	upyter 🛈 .3a4d27	running	Normal	registry-vpc.cn sha256:fdebb9458	192. :32768->8888/tcp 6006/tcp 192. :32769->22/tcp	17	192.	Delete   Stop   Monitor   Logs   Web Terminal
mydevbox_t 11ddefd632	ensorb 053030	running	Normal	registry-vpc.cn sha256:2b4c70877	192. :32770->6006/tcp	17	192	Delete   Stop   Monitor   Logs   Web Terminal

#### 手順 1: SSH トンネルのセットアップ

MAC OS X および Linux の場合

以下のコマンドを実行して、ローカルポートを ECS (Elastic Compute Service) インスタンス に接続します。

ssh - ND 12345 root @ 39 . 252

12345 は使用するローカルポートです。これはカスタマイズできます。 39 . 252 は、 ECS インスタンスにバインドされたパブリック EIP (Elastic IP) です。

# 🗎 注:

ECS コンソール にログインして ECS インスタンスにバインドされたパブリック EIP を確認でき ます。 ECS インスタンスにパブリック EIP がバインドされていない場合、 パブリック EIP をバ インドします。詳しくは、「パブリック EIP アドレスの管理」をご参照ください。

Instance List	China North 1	(Qingdao)	China N	orth 2 (Be	jing) China North 3 (Zha	angjiakou)	China East 1 (H	angzhou) China East	2 (Shanghai) China South	1 (Shenzhen)	0	Create Ins	tance
	Hong Kong	Asia Pacific	NE 1 (To	kyo) Asia	Pacific SE 1 (Singapore)	Asia Pacific	SE 2 (Sydney)	US East 1 (Virginia)	US West 1 (Silicon Valley)	Middle East 1 (Dubai)			
	EU Central 1 (F	Frankfurt)											
Instance Name	▼ Ent	er instance	e name (fu	zzy search	Search	🏶 Tag					Advanced Search	<u>a</u> (	?
Instance 1	ID/Name		Monitor	Zone	IP Address	Status(All	Network ) Type(All)	Configuration	VPC Details	Biling Method(All) 👻			Action
	786084 K34 84 738	<b>0</b> 0	R	China South 1 Zone B	39252(Elastic IP Address) 19287(Private IP Address)	Running	VPC	CPU: 4 Core(s) Memory: 30 GB (I/C Optimized) GPU: NVIDIA M40 100 Mbps ( peak value	vpc- vsw-	Pay-As- You-Go 17-06-28 15:46 created	Mar Change Confi	nage   Cor guration	nnect More <del>v</del>

Windows の場合

1. ローカルポート転送を設定します。

PuTTY を実行して SSH [トンネル] を設定します。

- a. [ソースポート] を設定します。 この例では、12345 となります。
- b. [ダイナミック] を選択します。
- c. [追加] をクリックします。

ategory:				
Session		Optio	ns controlling SSH p	ort forwarding
Logging     Logging     Terminal     Keyboard     Bell     Features     Window     Appearance     Behaviour		Port forwarding Local ports a Remote port Forwarded port	accept connections fr s do the same (SSH- s:	om other hosts 2 only) Remove
	Ш	Add new forward Source port Destination	ded port 12345	Add
Rlogin SSH		<ul> <li>Local</li> <li>Auto</li> </ul>	Remote	<ul> <li>Dynamic</li> <li>IPv6</li> </ul>
⊷ Kex	-			

2. ECS インスタンスにログインします。

- a. PuTTY を起動します。 [セッション] を設定して、 [開く] をクリックします。
  - IP アドレス、つまり ECS インスタンスのパブリック EIP を入力します。 この例では、 39 . 252 となります。

🔀 PuTTY Configuration		×					
Category:							
	Basic options for your PuTTY session						
	Specify the destination you want to connect to						
Keyboard	Host Name (or IP address)	Port					
Bell	39. 252	32769					
· · · · · · · · · · · · · · · · · · ·	Connection type: Raw Telnet Rlogin SSH	🔘 Serial					
Behaviour     Translation     Selection     Colours     Connection     Data     Proxy	Load, save or delete a stored session Saved Sessions Default Settings	Load					
Telnet Rlogin ⊞- SSH Serial		Delete					
	Close window on exit: Always Never Only on cle	an exit					
About	Open	Cancel					

b. 表示されたダイアログボックスで、ログインアカウントおよび ECS インスタンスのパス ワードを入力します。

これで、ECS インスタンスに正常にログインしました。

![](_page_26_Picture_2.jpeg)

#### 手順 2: ブラウザのネットワーク接続の設定

#### Firefox の場合

Firefox ブラウザを使用する場合、ブラウザを開き、[ツール] > [オプション] > [高度な設定] > [ネットワーク] をクリックして、 [接続] セクションの [設定] をクリックします。 表示されたウィンドウで、[SOCKS ホスト] を設定します。

![](_page_26_Picture_6.jpeg)

	Connection Settings			×
Configure Provies	to Access the Internet			
	o Access the internet			
Auto-detect pro	oxy settings for this net <u>w</u> ork			
Use system pro	xy settings			
Manual proxy c	onfiguration:			
HTTP Proxy:		Port:	0	•
5	Use this proxy server for all protocols			
SS <u>L</u> Proxy:		P <u>o</u> rt:	0	• ~
ETP Proxy:		Po <u>r</u> t:	0	•
SO <u>C</u> KS Host:	localhost	Por <u>t</u> :	12345	
	SOC <u>K</u> S v4 💿 SOCKS <u>v</u> 5			
<u>N</u> o Proxy for:				
localhost, 12	7.0.0.1			
Example: .mozi	lla.org, .net.nz, 192.168.1.0/24			
Automatic prox	y configuration URL:			
			R <u>e</u> loa	d
Do not prompt	for authentication if password is saved			
Proxy DNS when	n using SOCKS v5			
	бок	Cancel	<u>H</u> elp	C

#### Chrome の場合

以下のコマンドを実行します。

```
Chrome -- proxy - server =" socks5 :// localhost : 12345 " -- host -
resolver - rules =" MAP * 0 . 0 . 0 . 0 , EXCLUDE localhost "
-- user - data - dir =/ tmppath /
```

説明

- ・ 12345 は使用するローカルポートです。
- ・Windows では、/ tmppath / は d :/ tmppath または同様のパスとして記載されま
  - す。 Linux または MAC OS X では、/tmppath/ は / tmp / として記載されます。

Chrome の場所はオペレーティングシステムによって異なり、以下の表のようになります。

オペレーティングシステム	Chrome の場所
Mac OS X	/ Applicatio ns / Google \ Chromapp / Contents / MacOS / Google \ Chrome
Linux	/ usr / bin / google - chrome
Windows	C :\ Program Files ( x86 )\ Google \ Chrome \ Applicatio n \ chrome . exe

Windows を例にとります。

C:\U€	ers	>"C:\Program	Files (x8	6)\Google\	Chrome\App	lication∖	chrome	e.ex
e" ·	-proxy-server="	socks5://loca	1host:123	45"host	-resolver-	rules="MA	P × 0.	.0.0
.0,	EXCLUDE localho	st"user-da	ta-dir=d:	/tmppath				
C:∖U€	ers\	$\rangle$						

#### 手順 3: Jupyter サービスへのアクセス

お使いのブラウザで、Jupyter サービスのアクセスアドレスを入力します。 このページの例で は、アドレス 192. 83:32769 にアクセスします。 これにより、外部から SSH により Jupyter サービスにアクセスできます。

#### Firefox の場合

😜 🔤 🕺 🗙 🔿 Jupyter Notebook	× +		
/login?next=%2Ftree%3F	<b>V</b>	嬲 ▼ C Q	☆自
Banne   rearrest   and a land			
💭 Jupyter			
	Password:	Log in	
Chrome の場合			
C Jupyter Notebook ×			
$\leftrightarrow$ $\rightarrow$ C (192.	/login?next=%2Ftre	e%3F	
💭 Jupyter			

Password:

Log in

# 5 モデルトレーニング

# 5.1 スタンドアロンモデルトレーニング

Alibaba Cloud により提供されるエラスティックコンピューティングリソースおよびストレージ サービスを使用してモデルトレーニングコードを実行すると、スタンドアロントレーニングの繰 り返しを素早く行うことができます。 トレーニング中、いつでもログの確認およびトレーニング ステータスのモニタリングが可能です。

#### 前提条件

モデルトレーニングタスクの実行前に、以下の操作を実行したことを確認します。

- 一定数のエラスティックコンピューティングリソース (ECS (Elastic Compute Service) また は EGS) を含むコンテナークラスターを作成します。詳しくは、コンテナークラスターの作 成をご参照ください。
- モデルトレーニングに関するデータの保存に OSS (Object Storage Service) を使用するため、同一アカウントを使用して OSS バケットを作成します。次に、前述のコンテナークラスター上にデータボリュームを作成して、 OSS バケットをローカルディレクトリとしてトレーニングタスクを実行するコンテナーにマウントします。詳しくは、データボリュームの作成をご参照ください。

・モデルトレーニングコードを利用可能な GitHub コードリポジトリに同期させます。

規則

トレーニングコードでトレーニングデータの読み込み、トレーニングログの出力およびトレーニ ングの反復ステータスデータ (チェックポイント) の保存をするために、以下の規則に注意しま す。

- ・トレーニングデータは自動的に / input ディレクトリに保存されます。これは OSS 上のパスと一致します。コードはこのディレクトリからトレーニングデータを読み込みます。
- ・ログおよびチェックポイントファイルを含む、コードによって出力されるすべてのデータ
   は、/ output ディレクトリに格納されます。/ output ディレクトリに格納されるすべ
   てのファイルは、同じディレクトリ構造を持つ OSS バケットと自動的に同期されます。
- ・特別な Python 依存リポジトリがトレーニングコードに必要な場合、すべての依存関係を requiremen ts.txt 設定ファイルに書き込み、GitHub リポジトリのコードのルー トディレクトリに格納します。

#### 手順

- 1. Container Service コンソールにログインします。
- 左側のナビゲーションウィンドウから [イメージとテンプレート] > > [ソリューション]をク リックします。
- 3. [トレーニング]の[起動]をクリックします。

Container Service	Machine Learning		
Swarm Kubernetes			
Overview	D DevBox	Training	Prediction
Applications	Develop and debug models with Jupyter and	Train models on CPU, GPU with support for	Run prediction on CPU, GPU with support for
Services	Tensorboard. TensorFlow and Keras are supported.	TensorFlow and Keras. Visualize training with TensorBoard.	TensorFlow Serving. Load balancing and scalability are supported by nature.
Clusters	Launch   Guide	Launch Guide   History	Launch   Guide
Nodes		3	
Networks			
Data Volumes			
Configurations			
<ul> <li>Images and Tem</li> </ul>			
Docker Images			
Orchestration T			
Solutions 2			
Operation Logs			
Getting Started			

4. スタンドアロントレーニングタスクに関する基本情報を設定します。

以下の設定を完了させます。

Training			
	Cluster	EGS-cluster v	
	Application Name	tf-train-standalone	]
		The name should be 1-64 characters long, and can contain numbers, English letters and hyphens, but cannot start with a hyphen.	
ß	Framework	tensorflow:1.1.0	
		Distributed Training	
	GPUs Per Worker	0	
	Data Source	tfoss	
	Git URL	https://github.com/aymericdamien/TensorFlow-Examples	
		✓ Private Git Information	
	Git Username	Tanta	]
	Git Password	•••••	]
	Command	python <u>unsist_save.org</u> training_iteration=1000 data_ <u>dir=/invut/tersorflow/innist/</u> data log_ <u>dir=<sup>[</sup>output/training_logs]</u>	
		C Enable Monitor	
	Log Directory	/output/training_logs	]
		Please ensure the same log file directory is used in your code.	
		ок	

- ・クラスター:スタンドアロンモデルトレーニングタスクを実行するコンテナークラスターを 選択します。
- ・アプリケーション名: スタンドアロンモデルトレーニングタスクを実行するアプリケーションの名前で、1文字以上 64 文字以内で、数字、英字およびハイフン (-) を含むことができます。たとえば、"tf-train-standalone" となります。
- ・フレームワーク:モデルトレーニングに使用するフレームワークを選択します。現在、サポートされるフレームワークには、TensorFlow、Keras、Pythonおよびカスタマイズ イメージが含まれます。カスタマイズイメージを選択した場合、[イメージ]フィールドに 有効なイメージのアドレスを入力します。
- ・分散トレーニング:分散トレーニングタスクを実行するかどうかを選択します。ここでは、 スタンドアロントレーニングを実行するため、チェックボックスをオフにします。
- ワーカーごとの GPU: 使用する GPU 数を指定します。このパラメーターを "0" に設定すると、GPU の代わりに CPU を使用してトレーニングを行います。
- ・データソース:トレーニングデータの保存に使用するデータソースを選択します。 OSS に よりクラスター上に作成されたデータボリュームを選択するか、ローカルディレクトリを

- 選択して絶対パスを入力 入力します。 [データソースなし] を選択ですることもできます。 この例では、tfoss データボリュームを選択します。
- ・Git URL: トレーニングコードが保存されている GitHub コードリポジトリのアドレスを指 定します。

![](_page_32_Picture_4.jpeg)

現在、HTTP および HTTPS がサポートされますが、SSH はサポートされていません。

- ・プライベート Git 情報: プライベートコードリポジトリを使用する場合は、このチェック ボックスをオンにします。次に、Git ユーザー名 (お使いの GitHub アカウント) および Git パスワード を設定します。
- ・コマンド:モデルトレーニングを行う上記のコードの実行に使用するコマンドを指定します。

# 📃 注:

Python3 をサポートするフレームワークを選択した場合、 python の代わりに直接 python3 をコマンドラインで呼び出します。

 ・モニターの有効化: TensorBoard を使用してトレーニングデータをモニタリングするかど うかを選択します。このチェックボックスをオンにした場合、[ログディレクトリ] フィー ルドにトレーニングデータを保存する有効なパスを入力して、パスがトレーニングコード でのログ出力パスと同じものであることを確認します。たとえば、[ログディレクトリ]を / output / training\_l ogs に設定した場合、コードはログを同じパスに出力しま す。上の図に示すように、コードのコマンドラインパラメーター -- log\_dir の指定に よりパスの整合性が保証されます。

#### 注:

ここで、トレーニングログとは TensorBoard 用の TensorFlow API により出力された イベントファイル、およびモデルステータスを保存するチェックポイントファイルを示し ます。

5. 設定完了後、[OK] をクリックして、モデルトレーニングタスクの実行に使用するアプリケー ションを作成します。 左側のナビゲーションウィンドウから [アプリケーション] をクリックします。 [クラスター]
 ドロップダウンリストからクラスターを選択してから、作成したアプリケーションの名前 (この例では、tf-train-standalone) をクリックします。

Name	Description	Status	Container Status	Time Created 🔺	Time Updated 🔺	Action
tf-train-standalone		Ready	Ready:6 Stop:0	06/06/2018,15:24:07	06/06/2018,15:26:38	Stop   Update   Delete   Redeploy   Events

7. [ルート] タブをクリックします。 "tensorboard" から始まるリンクが表示されます。

Services	Containers	Logs	Events	Routes							
Route Address											
jerkins.cs2	astac76/ba454	804398	646542144	un-hengeh							

このリンクをクリックして TensorBoard モニタリングページを開きます。 トレーニングモニ タリングデータを参照できます。

$\leftarrow$ $\rightarrow$ $C$ $\square$ tensorboard-20170414100	823.c288	914c0cf6340	dcb98008e	e691e3ada8.c	n-shenzhen.alicontai	ner.com/?spm=517	76.2020520152.221.16.4Dx1HQ			<u>s</u> :
TensorBoard s	SCALARS	IMAGES	AUDIO	GRAPHS	DISTRIBUTIONS	HISTOGRAMS	EMBEDDINGS	C	۵	0
Write a regex to create a tag group	< 4	accuracy_1								1
Split on underscores		accuracy_1								
Data download links		0.900								
Tooltip sorting method: default	<b>*</b>	0.700								
		0.500								
Smoothing		0.300								
0.6		0.000	1.000k 2.00	0k 3.000k 4.000	k 5.000k					
		c) 🔳								
Horizontal Axis										
STEP RELATIVE WALL	(	cross_entropy	_1							1
	(	global_step								1
Runs	1	ayer_linear								8
Write a regex to filter runs										
<b>V</b> O ·										
✓ O test										
V (rain										
TOGGLE ALL RUNS										
/output/training_logs										

8. [ログ] タブをクリックすると、トレーニングタスクが実行中のときに、コードによっ てstdout/stderr に出力されるログが表示されます。

![](_page_33_Picture_9.jpeg)

は TensorBoard トレーニングモニタリングの実行に使用されます。 [サービス] タブまたは [コンテナー] タブをクリックして作成された サービスまたはコンテナーを表示できます。

たとえば、 tf - train - standalone \_worker\_1 のようにサービスコンテナー名で ログをフィルタリングでき、コンテナー操作プログラムにより stdout/stderr に出力された詳 細なログを参照できます。 時間および表示量によりログをフィルタリングすることもでき、ロ グファイルのローカルディレクトリへのダウンロードも選択できます。

🗎 注:

ここでの stdout/stderr ログは、上記の TensorBoard イベントファイルおよびチェックポ イントファイルとは異なります。

Services	Containers	Logs	Events	Routes							
Entries Por C						Filter by Container I		-	Filter by Start Time:		Developed Lane
Enules Per O	Intainer. 100it	ems v				Filter by Container I	All	*	Filter by Start Time.		Download Logs
tf-traim	-standalone_	_worker_1	2017-0	4-17T05:4	6:02.65836348	9Z Cloning trai	ning code fr	om ht	tp://www.github.c	com/wsxiaozhang/tf-m	nist.git
tf-train	-standalone_	_worker_1	2017-0	4-17T05:4	6:02.66247261	9Z Cloning into	'tf-mnist'.	••			
tf-train	-standalone_	_worker_1	2017-0	4-17105:4	6:07.67145310	22 Done cloning 07 Run training	code.	/start	er/tf_mnict act r	wthen mnist save ny	training ite
ration=1	000 data c	_worker_1 dir=/inpu	t/tensorf	low/mnist	/datalog d	ir=/output/trai	ning logs	/start	er/tr-mitst as: p	Jychon milist_save.py	training_ite
tf-train	-standalone	_worker_1	2017-0	4-17T05:4	6:11.05521961	8Z W tensorflow	/core/platfo	rm/cpu	_feature_guard.co	:45] The TensorFlow	library wasn't
compiled	to use SSE3	3 instruc	tions, bu	it these a	re available	on your machine	and could s	peed u	p CPU computation	15.	
tf-train	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.05527079	7Z W tensorflow	/core/platfo	rm/cpu	_feature_guard.co	:45] The TensorFlow	library wasn't
compiled	to use SSE4	1.1 instr	uctions,	but these	are availabl	e on your machi	ne and could	speed	up CPU computati	lons.	
tt-train	-standalone_	_worker_1	2017-0	4-17105:4	6:11.0552/932	/Z W tensortlow	/core/platto	rm/cpu	guard.co	:45] The TensorFlow	library wasn't
tf=trair	standalone	worker 1	1 2017_0	4_17T05+4	6+11_05528540	37 W tensorflow	/core/nlatfo	rm/cou	feature quard co		library wasn't
compiled	to use AVX	instruct	ions. but	these an	e available o	n vour machine	and could sp	eed up	CPU computations	he rensorr com	cibrary wash c
tf-train	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.05531332	0Z W tensorflow	/core/platfo	rm/cpu	_feature_guard.co	:45] The TensorFlow	library wasn't
compiled	to use AVX2	2 instruc	tions, bu	it these a	re available	on your machine	and could s	peed u	p CPU computation	15.	
tf-traim	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.05532317	3Z W tensorflow	/core/platfo	rm/cpu	_feature_guard.co	:45] The TensorFlow	library wasn't
compiled	to use FMA	instruct	ions, but	these ar	e available o	n your machine	and could sp	eed up	CPU computations		
tf-train	-standalone_	_worker_1	2017-0	4-17105:4	6:11.06822343	9Z WARNING:tens	orflow:From	mnist_	save.py:43: init:	lalize_all_variables	(from tensorfl
ow.pytho	on.ops.variat	worker 1	l 2017_0	a and will	L DE FEMOVED	atter 2017-03-0	for undatio				
tf-train	-standalone	worker 1	2017-0	4-17105:4	6:11.06824269	07 Use `tf.alob	al variables	initi	alizer` instead.		
tf-train	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.10538923	4Z WARNING:tens	orflow:From	mnist_	save.py:49: index	_to_string (from te	nsorflow.contri
b.lookup	.lookup_ops)	is depr	ecated an	d will be	removed afte	r 2017-01-07.		_			
tf-train	n-standalone_	_worker_1	2017-0	4-17T05:4	6:11.10539973	0Z Instructions	for updatin	g:			
tf-train	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.10540332	9Z This op will	be removed	after	the deprecation o	late. Please switch	to index_to_str
ing_tab	<pre>le_from_tenso</pre>	or and ca	ll the lo	okup meth	od of the ret	urned table.	-1				
tf-train	-standalone_	_worker_1	2017-0	4-17105:4	6:11.81693472	32 Training mod 07 Extraction /	el input/tensor	flow/m	nict/data/train_i	images_idv3_ubvte_gz	
tf-train	-standalone	worker 1	2017-0	4-17105:4	6:11.81696841	57 Extracting /	input/tensor	flow/m	nist/data/train_]	abels-idx1-ubvte.gz	
tf-train	-standalone_	worker_1	2017-0	4-17T05:4	6:11.81697204	9Z Extracting /	input/tensor	flow/m	nist/data/t10k-im	nages-idx3-ubyte.gz	
tf-traim	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.81697553	2Z Extracting /	input/tensor	flow/m	nist/data/t10k-la	abels-idx1-ubyte.gz	
tf-traim	-standalone	_worker_1	2017-0	4-17T05:4	6:11.81697897	7Z training acc	uracy 0.9092				
tf-train	-standalone_	_worker_1	2017-0	4-17T05:4	6:11.81698290	7Z Done trainin	g !				

9. [サービス] タブをクリックします。 ワーカーサービスのステータスが [停止] の場合、トレー ニングタスクが完了しています。

Services	Containers	Logs	Events	Routes				
Name		Applicatio	n		Status	Container Status	Image	Action
tensorboi	orboard tf-train-standalone		tf-train-standalone Ready Ready: Hegistry-vgc_tra-sherather, silv_nct_com/searcher				Stop   Restart   Reschedule   Update   Delete   Events	
worker		tf-train-st	andalone		Stopped	Ready:0 Stop:1	registry-spir.co-divertilentally.acc.com/ansarfan	Activate   Restart   Reschedule   Update   Delete   Events

10.OSS クライアントで、OSS バケットに自動的に格納されたトレーニング結果を確認できま す。

トレーニング完了後、ローカルの / output ディレクトリに格納されたすべてのファイル が、システムにより自動的に指定したデータボリュームに対応する OSS バケットにコピーさ れます。OSS クライアントでは、/ output ディレクトリに書き込まれたイベントファイ ルおよびチェックポイントファイルが OSS バケットにバックアップされたことを 確認できま す。

0	tensorflow-samples								
Obj	ect Management				♥ Upload + Create Folder	Refresh	Input object key prefix		Q Search
	Folder Name	Size	Туре		Creation Time				Action
Î	training_logs/ Go back up a level								
	test/			Folder					Delete
	train/		-	Folder	-				Delete
	checkpoint		0.155KB	Standard	2017-07-04 18:12:22		Get URL	Set HTTP Header	Delete
	ckpt.data-00000-of-00001		165.05KB	Standard	2017-07-04 18:12:07		Get URL	Set HTTP Header	Delete
	ckpt.index		196.478KB	Standard	2017-07-04 18:12:07		Get URL	Set HTTP Header	Delete
	.ckpt.meta		931.77KB	Standard	2017-07-04 18:12:19		Get URL	Set HTTP Header	Delete
\$	elect All Cancel Selection Batch Delete Batch Sel	t HTTP Header				Fo	r more flexible operations, try th	e OSS client tools:	Win   Mac

11.トレーニングタスクを管理します。

トレーニングタスクの停止、再起動または削除するには (たとえば、新しいトレーニングタス クのためにコンピューティングリソースをリリースするため)、左側のナビゲーションウィン ドウから [アプリケーション] をクリックして アプリケーションリストページに戻ります。対 応するアプリケーションを検索して、右側にある操作を実行します。

Cluster: EG	GS-cluster 🔻	✓ Hide System Applications □ Hide Offline	Hide System Applications 🔠 Hide Offline Applications 🔤 Hide Online Applications								
Name		Description	Status	Container Status	Time Created 🔺	Time Updated 🔺	Action				
tf-train-stand	idalone	eml solution application	Ready	Ready:1 Stop:1	2017-07-04 14:49:11	2017-07-04 14:50:59	Stop   Update   Delete   Redeploy   Events				

ここで説明しているモデルトレーニングサービスを使用することで、最初からモデルをト レーニングすることができるだけでなく、新しいデータを使用して既存のモデル (チェックポ イント)を基にしたトレーニング (たとえば、微調整)を継続することもできます。既存のア プリケーションの設定を継続的に更新することで、ハイパーパラメーターを調整し、反復的 なトレーニングを実行できます。

# 5.2 分散モデルトレーニング

Alibaba Cloud により提供されるエラスティックコンピューティングリソースおよびストレージ サービスを使用して分散トレーニングを迅速に行うことで、モデルトレーニングコードを実行で きます。 トレーニング中、コンピューティングリソース (CPU および GPU) の分配方法の制御、 ログの確認、およびトレーニングステータスのモニタリングをいつでも行え、トレーニング結果 をストレージサービスにバックアップできます。 このドキュメントで説明するモデルトレーニングサービスを使用することで、最初からモデルを トレーニングするだけでなく、新しいデータを使用して既存のモデル (チェックポイント) に基づ いたトレーニングの継続 (たとえば、微調整) も 行えます。 既存のアプリケーションの設定を継 続的に更新することで、ハイパーパラメータを調整し、反復的なトレーニングを実行できます。

#### 前提条件

モデルトレーニングタスクの実行前に、以下の操作を実行したことを確認します。

- 一定数のエラスティックコンピューティングリソース (ECS (Elastic Compute Service) または EGS) を含むコンテナークラスターを作成します。詳しくは、コンテナークラスターの作成をご参照ください。
- モデルトレーニング用データの保存に OSS (Object Storage Service) を使用するために、同 じアカウントで OSS バケットを作成します。次に、前述のコンテナークラスターでデータボ リュームを作成し、トレーニングタスクを実行するコンテナーにローカルディレクトリとして OSS バケットをマウントします。詳しくは、データボリュームの作成をご参照ください。
- ・モデルトレーニングコードを利用可能な GitHub コードリポジトリと同期させます。

規則

トレーニングコードでトレーニングデータを読み込み、トレーニングログを出力し、トレーニン グの反復ステータスデータ (チェックポイント) の保存を簡単にするために、以下の規則にご注意 ください。

- ・トレーニングデータは自動的に / input ディレクトリに保存されます。これは OSS 上のパスと一致します。コードはこのディレクトリからトレーニングデータを読み込みます。
- ログおよびチェックポイントファイルを含む、コードによって出力されるすべてのデータは、/ output ディレクトリに保存されます。/ output ディレクトリに保存されるすべてのファイルは、同じディレトリ構造を持つ OSS バケットに 自動的に同期されます。特別な Python 依存リポジトリがトレーニングコードに必要な場合、すべての依存関係をrequiremen ts.txt 設定ファイルに書き込み、そのファイルを GitHub リポジトリのコードルートディレクトリに格納します。

#### 手順

- 1. Container Service コンソールにログインします。
- Swarm で、左側のナビゲーションウィンドウから [イメージとテンプレート] > > [ソリューション] をクリックします。

### 3. [トレーニング] の [起動] をクリックします。

Containe	er Service	Machine Learning		
Swarm	Kubernetes			
Overvie	w	DevBox	Training	P Prediction
Applicat Services	ions	Develop and debug models with Jupyter and Tensorboard. TensorFlow and Keras are supported.	Train models on CPU, GPU with support for TensorFlow and Keras. Visualize training with TensorBoard.	Run prediction on CPU, GPU with support for TensorFlow Serving. Load balancing and scalability are supported by nature.
Clusters		Launch   Guide	Launch Guide   History	Launch   Guide
Nodes Network	IS		3	
Configu	rations			
<ul> <li>Images</li> <li>Docker</li> </ul>	and Tem Images	1		
Orches Solutio Operatio	tration T ns 2 on Logs			
Getting	Started			

4. 分散トレーニングタスクに関する基本情報を設定します。

以下の設定を完了させます。

Training	
Cluster	EGS-cluster v
Application Name	tf-train-distributed
	The name should be 1-64 characters long, and can contain numbers, English letters and hyphens, but cannot start with a hyphen.
Framework	tensorflow:1.1.0 v
	Distributed Training
	Number of Parameter Servers 1
	Number of Workers 2
GPUs Per Worker	1
Data Source	tfoss •
Git URL	https://github.com/wsxiaozhang/tf-mnist
	Private Git Information
Command	python <u>mnist_dist_train.ny</u> data_dir=/input/tensorflow/mnist/data/ train_steps=5000log_device_placement=False [log_dir=/output/mnist/dist/log-] batch_size=100learning_rate=0.01sync_replicas=False <u>num_gpus</u> =2
	✓ Enable Monitor
Log Directory	/output/mnist/dist/log Please ensure the same log file directory is used in your code.
	ОК

- クラスター:分散モデルトレーニングタスクを実行するコンテナークラスターを選択します。
- ・アプリケーション名: トレーニングタスクの実行用に作成するアプリケーションの名前は、
   1 文字以上 64 文字以内で、数字、英字およびハイフン (-) を含むことができます。 たとえば、"tf-train-distributed" となります。
- ・フレームワーク:モデルトレーニングに使用するフレームワークを選択します。現在、サポートされるフレームワークには、TensorFlow、Keras、Pythonおよびカスタマイズ イメージが含まれます。カスタマイズイメージを選択した場合、[イメージ]フィールドに 有効なイメージのアドレスを入力します。
- 分散トレーニング:分散トレーニングを実行するかどうかを選択します。ここでは、分散トレーニングを実行するため、チェックボックスをオンにします。パラメーターサーバー数およびワーカー数を設定し、分散トレーニング中の計算に使用するパラメーターサーバーノード数およびワーカーノード数を指定します。この例では、1つのパラメーターサーバーと2つのワーカーを設定します。
- ・ ワーカーごとの GPU: トレーニングタスクが実行される際のそれぞれのワーカーに対して
   使用する GPU 数を指定します。このパラメーターを "0" に設定すると、GPU の代わり

に CPU を使用してトレーニングを行います。 この例では、それぞれのワーカーで 1 つの GPU を使用します。

- ・データソース: トレーニングデータの保存に使用するデータソースを選択します。 クラス ターで OSS により作成されたデータボリュームまたはローカルディレクトリを選択し、 絶対パスを入力します。 [データソースなし] を選択ですることもできます。 この例で は、"tfoss" データボリュームが選択されてます。
- ・ Git URL: トレーニングコードが格納されている GitHub コードリポジトリのアドレスを指 定します。

現在、HTTP および HTTPS がサポートされますが、SSH はサポートされていません。 ・プライベート Git 情報: プライベートコードリポジトリを使用する場合は、このチェック ボックスをオンにします。次に、Git ユーザー名 (お使いの GitHub アカウント) および Git パスワードを設定します。

・コマンド:モデルトレーニングを行う上記のコードの実行に使用するコマンドを指定します。

**注**:

Python3 をサポートするフレームワークを選択した場合、 python の代わりに直接 python3 をコマンドラインで呼び出します。

・モニターを有効にする: トレーニングステータスをモニターするために TensorBoard を使用するかどうかを選択します。このチェックボックスにチェックを入れて、[ログディレクトリ] フィールドにトレーニングデータを格納する有効なパスを入力して、そのパスがトレーニングコードのログ出力パスと同じものであることを確認します。たとえば、[ログディレクトリ]を / output / training\_l ogs に設定した場合、コードはログを同じパスに出力します。上の図に示すように、パスの整合性はコードのコマンドラインパラメーター -- log\_dir を指定することで保証されます。

/ \_ \_ 注:

ここで、トレーニングログとは TensorBoard 用の TensorFlow API により出力された イベントファイル、およびモデルの状態を格納するチェックポイントファイルを示して います。(TensorFlow を利用した分散トレーニングの実行時、主要なロールを行うワー カーが、通常チェックポイントを保存するロールを担います)。

5. 設定完了後、[OK] をクリックして、モデルトレーニングタスクの実行に使用するアプリケー ションを作成します。 左側のナビゲーションウィンドウから [アプリケーション] をクリックします。 [クラスターリスト] からクラスターを選択して、作成したアプリケーションの名前をクリックします (この 例では tf-train-distributed となります)。

Help: & Create an application & Change application configurations & Simple route blue-green release policy & Container auto scaling										
Cluster: EGS-cluster 🔻 🗭 Hide System Applications 🔲 Hide Offline Applications 🔛 Hide Online Applications										
Name	Description	Status	Container Status	Time Created 🔺	Time Updated 🔺	Action				
tf-train-distributed	eml solution application	Ready	Ready:3 Stop:2	2017-07-04 18:07:46	2017-07-04 18:11:29	Stop   Update   Delete   Redeploy   Events				

[ルート] タブをクリックします。 "tensorboard" で始まる2つのリンクが表示されます。 それぞれのリンクを使用して対応するワーカーノードの TensorBoard モニタリングページを開きます。 "tensorboard0" で始まるリンクはワーカー "0" を示し、"tensorboard1" で始まるリンクはワーカー "1" を示します。

Services	Containers	Logs	Events	Routes		
Route Addre	ISS					
tensorboard0-20170704180746						
tensorboard	1-201707041807	46				

#### リンクをクリックしてトレーニングモニタリングデータを表示します。

$\leftrightarrow$ $\rightarrow$ $\subset$ $\bigcirc$ tensorboard0-201704	14100823.c	288914c0cf63	40dcb980	008e691e3ad	a8.cn-shenzhen.alico	ontainer.com/?spm=	=5176.2020520152.221.17.VdajsK		\$
TensorBoard	SCALARS	IMAGES		GRAPHS				G	۵
Write a regex to create a tag group	×	accuracy_1							
Split on underscores		accuracy_1							
Data download links		0.900	<u></u>						
Tooltip sorting method: default	~	0.700							
		0.500							
Smoothing		0.100							
<u>0</u> .	6	0.0	00 1.000k 2	.000k 3.000k 4.0	100k 5.000k				
		C 🔳							
Horizontal Axis									
STEP RELATIVE WA	ALL	cross_entrop	oy_1						
		layer_linear							

8. [ログ] タブをクリックして、トレーニングタスクの実行時にお使いのコードにより stdout/ stderr に出力されたログを表示します。

🗎 注:

通常、トレーニングタスクアプリケーションの実行のために、複数のサービスコンテナーが 自動的に作成され、それぞれ異なるプログラムブランチを実行します。 たとえば、"ps" コン テナーはパラメーターサーバーコードの実行、 "woker" コンテナーはモデルコンピューティ ングコードの実行、"tensorboard" コンテナーは TensorBoard トレーニングモニタリング の実行に使用されます。 [サービス] タブまたは [コンテナー] タブをクリックして作成された サービスまたはコンテナーを 表示できます。

たとえば、"worker0" のようにサービスコンテナー名でログをフィルタリングでき、コンテ ナー操作プログラムにより stdout/stderr に出力された詳細なログを参照できます。 時間と 表示量によりログをフィルタリングすることもでき、ログファイルのローカルディレクトリへ のダウンロードも選択できます。 この例では、"worker0" および "worker1" のログの確認 後、"worker0" と "worker1" が一部のトレーニングデータに基づいた並列モデルにより同じ モデルを計算し、"ps0" を利用して同時に勾配の更新を行なっていることを確認できます。

# 🗎 注:

ここで説明した stdout/stderr ログは、前述の TensorBoard イベントファイルおよび チェックポイントファイルとは異なります。

#### "worker0" ログ

Services	Containe	rs Logs	Events	Routes									
												1	
Entries Per C	Container: 20	00items 🔹								Filter by Container Name	worker0 🔻	Filter by Start Time:	Download L
												·	
workerØ	2017-07-	04T10:12:05	.213454241	Z 1499163124.8464	46: Worker 0:	training ste	p 2713 done	(global ste	: 4753)				
workerØ	2017-07-	04T10:12:05	.213457040	Z 1499163124.8553	96: Worker 0:	training ste	p 2714 done	(global ste	: 4754)				
workerØ	2017-07-	04T10:12:05	.213459832	Z 1499163124.8623	14: Worker 0:	training ste	p 2715 done	(global ste	: 4755)				
workerØ	2017-07-	04T10:12:05	.213465963	Z 1499163124.8699	52: Worker 0:	training ste	p 2716 done	(global step	: 4756)				
workerØ	2017-07-0	04T10:12:05	.213468881	Z 1499163124.8773	07: Worker 0:	training ste	p 2717 done	(global ste	: 4757)				
workerØ	2017-07-0	04T10:12:05	.213471677	Z 1499163124.8852	65: Worker 0:	training ste	p 2718 done	(global ste	: 4758)				
workerØ	2017-07-0	04T10:12:05	.213474496	Z 1499163124.8922	12: Worker 0:	training ste	p 2719 done	(global ste	: 4759)				
workerØ	2017-07-0	04T10:12:05	.213477291	Z 1499163124.8988	61: Worker 0:	training ste	p 2720 done	(global ste	: 4760)				
workerØ	2017-07-0	04T10:12:05	.213480114	Z Accuracy at loc	al step 2721:	0.9668							
workerØ	2017-07-0	04T10:12:05	.213482816	Z 1499163125.0008	65: Worker 0:	training step	p 2722 done	(global ste	: 4770)				
workerØ	2017-07-	04T10:12:05	.213485605	Z 1499163125.0080	68: Worker 0:	training step	p 2723 done	(global step	: 4771)				
worker0	2017-07-	04T10:12:05	.213488429	Z 1499163125.0148	82: Worker 0:	training step	p 2724 done	(global step	: 4772)				
workerØ	2017-07-0	04110:12:05	.213491093	2 1499163125.0216	/9: Worker 0:	training step	p 2725 done	(global ste	: 4773)				
workerØ	2017-07-0	84110:12:05	.213493885	2 1499163125.0290	33: Worker 0:	training step	p 2726 done	(global ste	: 4774)				
workerØ	2017-07-0	04110:12:05	.213496/14	2 1499163125.03/1	bb: Worker Ø:	training step	p 2/2/ done	(global step	(: 4775)				
workerø	2017-07-	04110:12:05	.213499502	2 1499165125.0451	/0: Worker 0:	training ste	p 2728 done	(global ste	(: 4776)				
workerø	2017-07-	04110:12:05	.213502108	2 1499163125.0523	14: WOFKEF 0:	training ste	p 2729 done	(global ste	(14777)				
workerø	2017-07-0	04110:12:03	.213504662	7 1499165125.0590 7 Assumption at les	ээ: worкer 0:	craining ste	p 2730 done	(giobai ste	(: 4778)				
workero	2017-07-	04110:12:03	.213511455	7 140016212E 1612	75. Neekee 0.	training sto	a 2722 dana	(alabal sta	4797)				
workero	2017-07-	04110:12:03	213514207	7 1400162125.1013	75: Worker 0:	training step	2732 done	(global step	(1 4707)				
worker@	2017-07-0	04110:12:03 04T10:12:05	2135108957	7 1499163125.1085	P3: Worker 0:	training ste	2733 done	(global_ste	4709)				
worker@	2017-07-0	94710-12-05	213522475	7 1400163125 1833	51: Worker 0:	training ste	2735 done	(global ste	4791)				
workerØ	2017-07-	94T10-12-05	213525235	7 1499163125 1901	72: Worker 0:	training ste	2736 done	(global ste	4792)				
workerØ	2017-07-	04T10:12:05	.213528034	7 1499163125.1980	73: Worker 0:	training ste	2737 done	(global ste	: 4793)				
workerØ	2017-07-	04T10:12:05	.213530898	z 1499163125.2061	35: Worker 0:	training ste	2738 done	(global ste	: 4794)				
workerØ	2017-07-	04T10:12:00	.165304540	z 1499163125.2131	31: Worker 0:	training ste	2739 done	(global ste	: 4795)				
workerØ	2017-07-	04T10:12:00	.165341380	Z 1499163125.2215	22: Worker 0:	training ste	p 2740 done	(global ste	: 4796)				
worker0	2017-07-0	04T10:12:00	.165346507	Z Accuracy at loc	al step 2741:	0.9672							
worker0	2017-07-	04T10:12:00	.165349793	z 1499163125.3273	94: Worker 0:	training ste	p 2742 done	(global ste	: 4806)				
workerØ	2017-07-	04T10:12:00	.165352753	z 1499163125.3346	91: Worker 0:	training ste	p 2743 done	(global ste	: 4807)			A	

#### "worker1" ログ

Entries Per Container: 2000:ems • worker1 2817-07-04718:12:04.6719184912 149 worker1 2817-07-04718:12:04.6719212072 149 worker1 2017-07-04718:12:04.6719212072 149 worker1 2017-07-04718:12:104.6719293472 149 worker1 2017-07-04718:12:104.6719393472 149 worker1 2017-07-04718:12:104.6719393472 149 worker1 2017-07-04718:12:104.6719347302 149 worker1 2017-07-04718:12:04.6719347302 149 worker1 2017-07-04718:12:04.6719347302 149 worker1 2017-07-04718:12:04.6719347302 149 worker1 2017-07-04718:12:04.6719347302 149 worker1 2017-07-04718:12:04.6719347302 149 worker1 2017-07-04718:12:04.6719597682 149 worker1 2017-07-04718:12:04.6719597682 149 worker1 2017-07-04718:12:04.6719597682 149 worker1 2017-07-04718:12:04.6719597682 149	utes		
<pre>worker1 2017-07-04T10:12:04.6719184912 149 worker1 2017-07-04T10:12:04.671912407 149 worker1 2017-07-04T10:12:04.671923972 Acct worker1 2017-07-04T10:12:04.671932972 Acct worker1 2017-07-04T10:12:04.6719329622 149 worker1 2017-07-04T10:12:04.6719329621 149 worker1 2017-07-04T10:12:04.671932902 149 worker1 2017-07-04T10:12:04.671934921 149 worker1 2017-07-04T10:12:04.671934921 149 worker1 2017-07-04T10:12:04.671934921 149 worker1 2017-07-04T10:12:04.6719530632 149 worker1 2017-07-04T10:12:04.6719530632 149 worker1 2017-07-04T10:12:04.6719530632 149 worker1 2017-07-04T10:12:04.6719530632 149 worker1 2017-07-04T10:12:04.671951012 Acc worker1 2017-07-04T10:12:04.671951012 140 worker1 2017-07-04T10:12:04.671952052 149</pre>		Filter by Container Name worker1 • Filter by Start Time:	Download Logs
uorkeri 2017-07-04T19:12:04.671954582 109 uorkeri 2017-07-04T19:12:04.6719623421 109 uorkeri 2017-07-04T19:12:04.671970871 109 uorkeri 2017-07-04T19:12:04.671970872 109 uorkeri 2017-07-04T19:12:04.671970872 109 uorkeri 2017-07-04T19:12:04.6719708702 109 uorkeri 2017-07-04T19:12:04.6719987021 109 uorkeri 2017-07-04T19:12:04.6719897257 109 uorkeri 2017-07-04T19:12:04.6719897257 109 uorkeri 2017-07-04T19:12:04.6719897257 109	P015114004665: Worker 1: training step 2519 done (global step: 4660) P015124.012804: Worker 1: training step 2520 done (global step: 4661) P015124.012804: Worker 1: training step 2522 done (global step: 4671) P015124.114569: Worker 1: training step 2523 done (global step: 4672) P015124.114569: Worker 1: training step 2525 done (global step: 4673) P015124.12969: Worker 1: training step 2525 done (global step: 4676) P015124.13960: Worker 1: training step 2525 done (global step: 4676) P015124.13960: Worker 1: training step 2525 done (global step: 4676) P015124.13960: Worker 1: training step 2525 done (global step: 4676) P015124.149769: Worker 1: training step 2529 done (global step: 4679) P015124.12950: Worker 1: training step 2529 done (global step: 4679) P015124.23230: Worker 1: training step 2530 done (global step: 4679) P015124.23230: Worker 1: training step 2530 done (global step: 4679) P015124.23230: Worker 1: training step 2530 done (global step: 4679) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.23230: Worker 1: training step 2530 done (global step: 4680) P015124.2	Hiter by Container Names worker1 • Hiter by Start Lime:	Download Logs
worker1   2017-07-04T10:12:04.6719926802 1495 worker1   2017-07-04T10:12:04.6719953312 1495 worker1   2017-07-04T10:12:04.6719983332 1495	9163124.457956: Worker 1: training step 2543 done (global step: 4708) 19163124.466050: Worker 1: training step 2544 done (global step: 4709) 19163124.473918: Worker 1: training step 2545 done (global step: 4710)		

### 9. [サービス] タブをクリックします。 すべてのワーカーサービスのスーテタスが [停止] の場 合、トレーニングタスクが完了しています。

-	_											
Services	s Cor	ontainers	Logs	Events	Routes							
Name		1	Application			Status	Container Status	Image				Action
ps0		t	tf-train-dist	ributed		Ready	Ready:1 Stop:0	registry-vpc.cn-shenzhen.allyuncs.com/tensorflow	Stop	Restart	Reschedule Delete	Update     Events
tensorboa	ard0	t	tf-train-dist	ributed		Ready	Ready:1 Stop:0	registry-vpc.cn-shenzhen.aliyuncs.com/tensorflow	Stop	Restart	Reschedule Delete	Update     Events
tensorboa	ard1	t	tf-train-dist	ributed		Ready	Ready:1 Stop:0	registry-vpc.cn-shenzhen.aliyuncs.com/tensorflow	Stop	Restart	Reschedule Delete	Update     Events
worker0		t	tf-train-dist	ributed		Stopped	Ready:0 Stop:1	registry-vpc.cn-shenzlon.aliyuncs.com/tensorflow	Activate	Restart	Reschedule Delete	Update     Events
worker1		t	tf-train-dist	ributed		Stopped	Ready:0 Stop:1	registry-vpc.cn-shenzhen.aliyuncs.com/tensorflowvcm/tensorflow-amples/tensorflo w-train:1.1.0-devel	Activate	Restart	Reschedule Delete	Update     Events

10.OSS クライアントでは、OSS バケットに自動的に格納されたトレーニング結果を確認できま す。

トレーニング完了後、ワーカーコンテナーのローカル / output ディレクトリに保存された すべてのファイルが、システムにより自動的に指定したお使いのデータボリュームに対応する OSS バケットにコピーされます。OSS クライアントでは、 / output ディレクトリに書き 込まれたイベントファイルおよびチェックポイントファイルが OSS バケットにバックアップ されたことを 確認できます。

0	tensorflow-samples							
Obj	ect Management			Upload	Refresh	Input object key prefix	c	Search
	Folder Name	Size	Туре	Creation Time				Action
Ŷ	training_logs/ Go back up a level							
	test/		Folder					Delete
	train/	-	Folder					Delete
	events.out.tfevents.1499163079.fc901201eaa3-worker0	165.05KB	Standard	2017-07-04 18:12:07		Get URL Set HT	TP Header	Delete
	graph.pbbxt	196.478KB	Standard	2017-07-04 18:12:07		Get URL Set HT	TTP Header	Delete
	model.ckpt-0.data-00000-of-00001	931.77KB	Standard	2017-07-04 18:12:19		Get URL Set HT	TTP Header	Delete
	model.ckpt-0.index	0.556KB	Standard	2017-07-04 18:12:21		Get URL Set HT	TP Header	Delete
	model.ckpt-0.meta	86.151KB	Standard	2017-07-04 18:12:22		Get URL Set HT	TP Header	Delete
9	elect All Cancel Selection Batch Delete Batch Set HTTP Header				For	more flexible operations, try the OSS cl	ient tools: W	fin   Mac

11.トレーニングタスクを管理します。

トレーニングタスクの停止、再起動または削除するには (たとえば、新しいトレーニングタス クのためにコンピューティングリソースをリリースするため)、[アプリケーション] > [アプリ ケーションリスト] のページに戻ります。対応するアプリケーションを検索して、右側にある 操作を実行します。

# 6 予測

### 6.1 TensorFlow Serving の利用

TensorFlow Serving は、Alibaba Cloud エラスティックコンピューティングリソース (ECS (Elastic Compute Service) または EGS)、Server Load Balancer および OSS (Object Storage Service) を使用して TensorFlow モデルに関する予測を行います。

#### 前提条件

モデルトレーニングタスクの実行前に、以下の操作が実行されていることを確認します。

- 一定数のエラスティックコンピューティングリソース (ECS または EGS) を含むコンテナーク
   ラスターを作成します。詳しくは、コンテナークラスターの作成をご参照ください。
- ・同一アカウントを使用して、モデルトレーニング用のデータを保存する OSS バケットを作成 します。
- ・前述のコンテナークラスター用のデータボリュームを作成して、トレーニングタスクを実行するコンテナーにローカルディレクトリとして OSS バケットをマウントします。詳しくは、データボリュームの作成をご参照ください。
- TensorFlow Serving の基本概念と実行プロセスを理解します。詳しくは、『TensorFlow モデルの提供』をご参照ください。

#### 規則

モデル予測を簡素化するため、以下の規則に注意します。

- · OSS のルートディレクトリにモデル名を使用したフォルダーを作成すること。
- Server Load Balancer のフロントエンドとバックエンドのポートがモデル予測のアプリケーションポートと一致していることを確認する。

#### 手順

1. 共有ストレージに予測モデルを保存します。

OSS クライアント上のモデルフォルダーにアップロードすることにより完了させることができ ます。

a. OSS バケットのルートディレクトリにモデル名を使用したフォルダーを作成します。 この ページの例では、"mnist" を使用します。

o tensorflow-samples											
Object Management			♦ Upload	Refresh Input object key prefix	Q Search						
Folder Name	Size	Туре	Creation Time		Action						
📄 mnist/		Folder			Delete						
Select All Cancel Selection Batch Delete	Batch Set HTTP Header			For more flexible operations, try the OSS d	ient tools: Win   Mac						

b. TensorFlow Serving モデルフォルダーをバージョン番号を付けて "mnist" にアップロー ドします。

0	n tensorflow-samples												
Ob	ject Management			Upload     Create Folder	Refresh Input object key prefix	Q Search							
	Folder Name	Size	Туре	Creation Time		Action							
Ŷ	mnist/ Go back up a level												
	1/	-	Folder	-		Delete							
	Select All Cancel Selection Batch Delete	Batch Set HTTP Header			For more flexible operations, try the OSS clier	nt tools: Win   Mac							

- 2. Server Load Balancer のリスニングポートを設定します。
  - a. Server Load Balancer コンソールで、右上の [Server Load Balancer の作成] をクリック してルーティング用の Server Load Balancer インスタンスを作成します。

この例では、インターネット Server Load Balancer インスタンスを作成します。 ニーズ に応じて、インターネット Server Load Balancer インスタンスまたはイントラネット Server Load Balancer インスタンスの作成を選択できます。

![](_page_44_Picture_12.jpeg)

使用する Container Service クラスターと同じリージョンを選択します。これ は、Server Load Balancer がリージョン間のデプロイをサポートしていないためです。

b. Server Load Balancer コンソールに戻り、作成した Server Load Balancer インスタンスの名称を "TensorFlow-serving" と設定します。 Container Service は、この名前を使用して Server Load Balancer インスタンスを参照します。

左側のナビゲーションウィンドウから [インスタンス] をクリックします。 Server Load Balancer インスタンスが属するリージョンを選択します。 インスタンス名を編集して [確 認] をクリックします。

In	stance Management	China North 1 (Qingdao)	China North 2 (Beijing)	China North 3 (Zhangjiakou)	) China East 1 (Han	gzhou) China East 2 (Shang	hai) China South 1 (Shenzhen)	Hong Kong Asia Pacific NE 1 (T	okyo)	
		Asia Pacific SE 1 (Singapor	re) Asia Pacific SE 2 (S	Sydney) US East 1 (Virginia)	US West 1 (Silicon V	alley) Middle East 1 (Dubai)	EU Central 1 (Frankfurt)			
									Create Server	Load Balancer
Se	erver Load Balancer Name	Enter load b	alancer names separated	i by con Search 📎	Tag					<u>×</u> 0
	Server Load Balancer ID;	Name Zone	IP Address(All) 👻	Status Network(All) + I	Port/Health Check	Backend Server	Instance Spec	Bandwidth Billing Method(All) 👻	Billing Method(All) 👻	Action
	lb-wz9ryll0m (None)	cn-shenzhen-a(Masi cn-shenzhen-b(Slav	ter) 120.25.135.64 e) Public Network	Running Classic Network	Not ConfiguredConfigu	re Not ConfiguredConfigure	performance guaranteed instance slb.s1.small <b>()</b>	Pay by Traffic	Pay-As-You-Go 2017-07-04 19:09:10 Created	Manage   More 🗸
	Edit Server Load Bala	ncer Name :			Norm	c10a4679c453346 al c10a4679c453346 c10a4679c453346	performance guaranteed instance slb.s1.small 0	Pay by Traffic	Pay-As-You-Go 2017-07-04 17:38:05 Created	Manage   More 🗸
	It must be 1-80 chara	acters long. Only the letters a	a-z, numbers 0-9, and th	e characters '-' '/' ',' and '_' are a	allowed.	Not ConfiguredConfigure	performance guaranteed instance slb.s1.small ()	Pay by Traffic	Pay-As-You-Go 2017-07-04 16:33:26 Created	Manage   More 🗸
	Activate Stop	Release Settings	Edit Tags There	are 3 load balancers on the cur	rent node.					

c. リスニングポートを作成します。

インスタンスの右側にある [管理] をクリックします。 左側のナビゲーションウィンドウか ら [リスナー] をクリックします。 右上の [リスナーの追加] をクリックして、 設定を行いま

### す。 フロントエンドプロトコルとして TCP を選択し、ポートマッピングに "8000:8000" を設定します。

Add Listener		$\times$
1.Listener Configuration	2.Health Check > 3.Success	
Front-end Protocol [Port]:*	TCP : 8000 Port range is 1-65535.	
Backend Protocol [Port]:*	TCP         :         8000           Port range is 1-65535.	
Peak Bandwidth:	No Limits Configure Instances charged by traffic are not limited by peak bandwidth. Peak bandwidth range is 1-5000.	
Scheduling Algorithm:	Weighted F 🔻	
Use Server Group: 🕜		
Automatically Enable Listener After Creation:	Enable	
<ul> <li>Show</li> <li>Advanced</li> <li>Options</li> </ul>		
	Next Can	icel

- 3. モデル予測を開始します。
  - a. Container Service コンソールにログインします。
  - b. 左側のナビゲーションウィンドウから [イメージとテンプレート] > > [ソリューション]を クリックします。
  - c. [予測] の [起動] をクリックします。

Container Service	Machine Learning		
Swarm Kubernetes			
Overview	D DevBox	Training	Prediction
Applications	Develop and debug models with Jupyter and	Train models on CPU, GPU with support for	Run prediction on CPU, GPU with support for
Services	Tensorboard. TensorFlow and Keras are supported.	TensorFlow and Keras. Visualize training with TensorBoard.	TensorFlow Serving. Load balancing and scalability are supported by nature.
Clusters	Launch   Guide	Launch   Guide   History	Launch Guide
Nodes Networks Data Volumes Configurations III Images and Tem	1		3
Orchestration T Solutions Operation Logs			

d. モデル予測タスクに関する基本情報を設定します。

Prediction	
Cluster	EGS-cluster •
Application Name	tf-serving-dist
	The name should be 1-64 characters long, and can contain numbers, English letters and hyphens, but cannot start with a hyphen.
Framework	tensorflow-serving:0.5.1
Model Name	mnist
Number of Instances	1
GPUs Per Instance	0
Data Source	tfoss
Load Balancer Instance	TensorFlow-serving
Load Balancer Port	8000
	It should be used to set both frontend and backend ports of the specified load balancer.
	Ск

- ・クラスター:モデル予測を実行するクラスターを選択します。
- ・アプリケーション名: スタンドアロンモデル予測の実行に使用するアプリケーションの
   名前で、1文字以上 64 文字以内で、数字、英字およびハイフン (-) を含みます。
- ・フレームワーク:モデルトレーニングに使用するフレームワークを選択しま
  - す。 TensorFlow Serving およびカスタマイズイメージを含みます。 ここで
  - は、TensorFlow Serving を選択します。

- ・モデル名:ここでモデル名は手順1で作成したモデルフォルダー名と同じものである必要があります。
- インスタンス数: TensorFlow Serving インスタンス数です。これはクラスターでの ノード数を超えることはできません。
- ・インスタンスごとの GPU: 使用する GPU 数を指定します。このパラメーターを "0" に 設定すると、GPU の代わりに CPU を使用して予測を行います。
- ・データソース:予測モデルを保存するために OSS バケットによりクラスター上に作成されたデータボリュームを 選択します。データボリュームの作成方法については、データボリュームの作成ご参照ください。
- ・ロードバランサーインスタンス: 手順2で作成した Server Load Balancer インスタン スを選択します。
- ・ ロードバランサーポート: 手順2で設定したポートを入力します。
- 4. 設定完了後、[OK] をクリックします。
- 5. アプリケーションリストページで、[クラスター] ドロップダウンリストからクラスターを選択 して、作成したアプリケーションの名称をクリックします。
- [ルート] タブをクリックします。Server Load Balancer が提供する端末アドレスが表示 されます。これにより、gRPC クライアントを使用して "Server Load Balancer アドレ ス:Server Load Balancer ポート" にアクセスできます。

Services	Containers	Logs	Events	Routes						
Route Address										
201.37.193.36										

# 7 TFRecord および HDFS による TensorFlow トレー ニングデータの準備

データの準備と前処理は、ディープラーニングおよびトレーニングプロセスにおいて重要な役割 を果たし、モデルトレーニングのスピードと品質に影響します。

TensorFlow は HDFS をサポートし、ビッグデータおよびディープラーニングを統合し、デー タの準備からモデルトレーニングまでのチェーンを完全なものにします。 Alibaba Cloud Container Service のディープラーニングソリューションは、 TensorFlow をサポートする3 つ の分散ストレージバックエンド (OOS (Object Storage Service)、NAS および HDFS) を提供し ています。

このドキュメントでは、データの TFRecord 形式への変換方法および、生成された TFRecord ファイルを HDFS に保存する方法を説明します。 この例では、 Alibaba Cloud E-MapReduce (Elastic MapReduce) の HDFS を使っています。

#### TFRecord を使用する理由

TFRecord は TensorFlow で定義される統一標準データフォーマットです。 マルチスレッド データの読み込みをサポートし、バッチサイズおよびエポックパラメーターを使用して、 シング ルのバッチサイズとトレーニングプロセス中の反復回数を制御します。 TFRecord はメモリー を効率よく使い、データの複製および移動を簡単に行えます。 そのため、TensFlow で大規模な ディープラーニングトレーニングを実行する 際の望ましいオプションとなります。

#### 手順 1: E-MapReduce クラスターの作成

E-MapReduce は Alibaba Cloud プラットフォームで動作するビッグデータ処理システムソ リューションです。詳しくは、『*E-MapReduce* の概要』をご参照ください。

E-MapReduce コンソールにログインして E-MapReduce クラスターを作成します。 E-

MapReduce クラスターの作成方法については、『*E-MapReduce クラスターの作成*』をご参照ください。

このページの例では、中国 (深セン) にあるクラスターが作成され、[ネットワークタイプ] が [VPC] に設定されます。

The myEMR Back to cluster list									
Cluster info		^							
ID/Name C-47PC/000000000000000000000000000000000000									
Region cn-shenzhen Current status Creating									
Start Time 2017/07/04 20:07:55 Running time 11second(s)									
Log function Open Log path cess://tensorflow-samples2	Log path oss://tensorflow-samples2								
Software configuration Bootstrap action/Software configuration normal									
High availability No ECS instance role AllyunEmrEcsDefaultRole									
Software information		^							
Product version EMR-3.2.0 Cluster type HADDOP									
Software information hive 2.0.1, nginx 1.10.2, spark 2.1.1, ganglia 3.7.2, tez 0.8.4, hdfs 2.7.2, hue 3.11.0, zeppelin 0.7.1, sqoop 1.4.6, yarn 2.7.2, pig 0.14.0									
Network information		^							
Network type VPC Select security group emr-default-security group (sg-wz9ariwt962ng1zxifj)									
Zone cn-shenzhen-b VPC/VSwitch vpc-wz96ybdsjijor29djdf4h / vsw-wz96z71x9k43b1z64mw8d									
MasterNode information		^							
Basic information 1 Bandwidth : 8M CPU : 4Core Memory : 16G Data disk configuration : SSD Cloud Disk 80G X 1 disk(s)									
ID Status Public IP (?) Private IP Hardware configuration									
I-wz94hmrbm6jyzk64x1lu         Initializing         192.168.1.102         CPU : 4Core   Memory : 166   Data disk configuration : SSD Cloud Disk   80G X 1 disk(s)									

#### 手順 2: コンテナークラスターの作成および 2 つのクラスター間のネットワークの統合

1. Container Service コンソールにログインして、同じ VPC (Virtual Private Cloud) 下に GPU コンテナークラスターを作成します。

Cluster List				You can crea	te up to 5 clus	sters and can ac	ld up to 20 nodes in	n each cluster.	Refresh	Create Cluster
Help: 🔗 Create cluster 🔗	How to add existing ECS instar	nces 🔗 Cross-zone n	ode management 🔗 Log S	ervice integra	tion 🔗 Conn	ect to cluster th	nrough Docker Clier	it		
Name 🔻										
Cluster Name/ID	Cluster Type	Region	Network Type	Cluster Status	Node Status 🕜	Number of Nodes	Time Created	Docker Version		Action
ElasticGPUService	Alibaba Cloud Cluster	China South 1 (Shenzhen)	VPC vpc-	Running	Healthy 🕽	2	2018-01-17 11:17:41	17.06.2-ce	Manage	View Logs   Delete Monitor   More -

- 2. ECS コンソールにログインして、Container Service クラスターのノードに E-MapReduce クラスターに対応するセキュリティグループを 追加します。
  - a. セキュリティグループの属するリージョン (このページの例では、中国 (深セン)) を選択し ます。 セキュリティグループの右側にある [インスタンスの管理] をクリックします。

							1		
Security Group List	China North 1 (Qingdao)	China North 2 (Beijing)	China North 3 (Zhangjiakou)	China East 1 (Hangzhou	) China East 2 (Shanghai	) China South 1 (Shenzhen)	Hong Kong	Asia Pacific NE 1 (Tokyo)	
	Asia Pacific SE 1 (Singapo	rre) Asia Pacific SE 2 (Sy	dney) US East 1 (Virginia)	US West 1 (Silicon Valley)	Middle East 1 (Dubai)	EU Central 1 (Frankfurt)			
									Create Security Group
Security Group ID 🔻	Enter security group ID	Se	earch 📎 Tag						<u>×</u> ?
Security Group ID	/Name VPC		Related Instances	Network Type	Created	Description	Tags		Action
sg-wz9ariwt962ng emr-default-securi	1izxifj vpc tygr	wz96ybdsjjior29djdf4h	3	VPC 2	2017-07-04 20:07:56	-		Modify   C	Ione Security Group Restore rules
sg-wz94hmrbm6jy alicloud-cs-auto-cr	nd4wj0h9 vpc	wz96ybdsjjior29djdf4h	3	VPC 2	2017-07-04 17:37:45	security group of ACS		Modify   C	Ione Security Group   Restore rules Manage Instances   Configure Rules
sg-wz9gubr4nl044 alicloud-cs-auto-cr	fm23aac vpc	wz96ybdsjjior29djdf4h	2	VPC 2	2017-07-04 16:33:07	security group of ACS		Modify   C	lone Security Group   Restore rules Manage Instances   Configure Rules

b. 右上の [ECS インスタンスの追加] をクリックします。 コンテナークラスターでノードを選択して、[OK] をクリックします。

Add an ECS Instance			$\times$
*Instance ID:	i-wz90see2i0kwox2ib1cs	•	
		ок	Cancel

#### 手順 3: TFRecord データの生成

このページの例では、モデルトレーニングサービスにより convert\_to \_records . py を実行する実行環境が提供され、TFRecord データが生成され、HDFS にデータが 保存されま す。

- 1. Container Service コンソールにログインします。
- 左側のナビゲーションウィンドウから [イメージとテンプレート] > [ソリューション] をク リックします。

3. [トレーニング] で [起動] をクリックします。

Container Service	Machine Learning		
Overview	D DevBox	Training	P Prediction
Applications	Develop and debug models with Jupyter and	Train models on CPU, GPU with support for	Run prediction on CPU, GPU with support for
Services	Tensorboard. TensorFlow and Keras are supported.	TensorFlow and Keras. Visualize training with TensorBoard.	TensorFlow Serving. Load balancing and scalability are supported by nature.
Clusters	Launch   Guide	Launch Guide History	Launch   Guide
Nodes		3	
Networks			
Data Volumes			
Configurations			
<ul> <li>Images and Tem</li> </ul>			
Docker Images			
Orchestration T			
Solutions 2			
Operation Logs			

4. モデルトレーニングの設定を行い、[OK] をクリックします。

このページの例での設定は以下のようになります。

- ・フレームワーク: tensorflow:1.0.0 を選択します。
- ・ワーカーごとの GPU: "0" を入力します。
- ・データソース: "データソースなし" を選択します。
- ・ Git URL: "https://code.aliyun.com/deeplearning/mnist-examples.git" を入力しま す。
- ・コマンド:

#### -- output / neural - style / output . jpg

Training * Back to Solution List	
Training	
Cluster	ElasticGPUService •
Application Name	prepare-data
	The name should be 1-64 characters long, and can contain numbers, Faulish letters and hyphens, but cannot start with a hyphen.
Framework	tensorflow:1.0.0 🔻
	Distributed Training
GPUs Per Worker	0
Data Source	No Data Source
Git URL	https://code.aliyun.com/deeplearning/mnist-exan
	Private Git Information
Command	python neural_ <u>style.or</u> iterations 50000content /neural- style/examples/1- <u>content.ipg</u> styles /neural-style/examples/1- <u>style.ipg</u> output /neural-style/ <u>output.ipg</u>
	Enable Monitor
	ОК

これにより、作成したアプリケーションが [アプリケーションリスト] ページに表示されます。 アプリケーション名をクリックします。 [ログ] タブをクリックして実行ログを参照します。こ れは、TFRecord ファイルが HDFS に保存されていることを示しています。

Entries Per Container: 100items V Filter by Container Name: All V Filter by Start Time: Download Logs prepare-data-12_worker_1   2017-05-23T11:33:02.718478793Z Cloning training code from https://code.aliyun.com/deeplearning/mnist-examples.c prepare-data-12_worker_1   2017-05-23T11:33:02.720364084Z Cloning into 'mnist-examples' prepare-data-12_worker_1   2017-05-23T11:33:07.340480568Z Done cloning code.	Services	Containers	Logs	Events	Routes						
<pre>prepare-data-12_worker_1   2017-05-23T11:33:02.718478793Z Cloning training code from https://code.aliyun.com/deeplearning/mnist-examples.g it prepare-data-12_worker_1   2017-05-23T11:33:02.720364084Z Cloning into 'mnist-examples' prepare-data-12_worker_1   2017-05-23T11:33:07.340480568Z Done cloning code.</pre>	Entries Per Co	ontainer: 100it	ems 🔻				Filter by Container Name	e: All 🔻	Filter by Start Time:		Download Logs
prepare-data-12_worker_1   2017-05-23T11:33:02.720364084Z Cloning into 'mnist-examples' prepare-data-12_worker_1   2017-05-23T11:33:07.340480568Z Done cloning code.	prepare-d it	lata-12_work	er_1	2017-05-2	23T11:33:	02.718478793	BZ Cloning trainin	g code from	https://code.ali	iyun.com/deeplear	ning/mnist—examples.g
prepare-data-12_worker_1   2017-05-23T11:33:07.340480568Z Done cloning code.	prepare-d	lata–12_work	er_1	2017-05-2	23T11:33:	02.720364084	IZ Cloning into 'm	nist-examples	····		
	prepare-d	lata–12_work	er_1	2017-05-2	23T11:33:	07.340480568	BZ Done cloning co	de.			
prepare-data-12_worker_1   2017-05-23T11:33:07.3405984112 Run training code under /starter/mnist-examples as: python convert_to_records.py directory hdfs://192.168.100.206:9000/mnist-tfrecord	prepare-d direct	lata–12_work ory hdfs://	er_1   192.168	2017-05-2	23T11:33: 9000/mni	07.340598413 st-tfrecord	LZ Run training co	de under /sta	rter/mnist-examp	oles as: python c	onvert_to_records.py
prepare-data-12_worker_1   2017-05-23T11:34:05.508679080Z Extracting MNIST_data/train-images-idx3-ubyte.gz	prepare-d	lata-12_work	er_1	2017-05-2	23T11:34:	05.50867908	Z Extracting MNIS	T_data/train-	images-idx3-ubyt	te.gz	
prepare-data-12_worker_1   2017-05-23T11:34:05.508706069Z Extracting MNIST_data/train-labels-idx1-ubyte.gz	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.508706069	Z Extracting MNIS	T_data/train-	labels-idx1-ubyt	te.gz	
prepare-data-12_worker_1   2017-05-23T11:34:05.508710856Z Extracting MNIST_data/t10k-images-idx3-ubyte.gz	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.50871085	SZ Extracting MNIS	T_data/t10k-i	.mages-idx3-ubyte	e.gz	
prepare-data-12_worker_1   2017-05-23T11:34:05.508714238Z <a href="https://www.example.gz">www.example.gz</a>	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.508714238	BZ Extracting MNIS	T_data/t10k-l	abels-idx1-ubyte	e.gz	
prepare-data-12_worker_1   2017-05-23T11:34:05.5087175902 Writing hdfs://192.168.100.206:9000/mnist-tfrecord/train.tfrecords	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.50871759	Z Writing hdfs://	192.168.100.2	06:9000/mnist-t1	frecord/train.tfr	ecords
prepare-data-12_worker_1   2017-05-23T11:34:05.5087208742 Writing hdfs://192.168.100.206:9000/mnist-tfrecord/validation.tfrecords	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.508720874	Z Writing hdfs://	192.168.100.2	06:9000/mnist-t1	frecord/validation	n.tfrecords
prepare-data-12_worker_1   2017-05-23T11:34:05.5087243722 Writing hdfs://192.168.100.206:9000/mnist-tfrecord/test.tfrecords	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.508724372	Z Writing hdfs://	192.168.100.2	06:9000/mnist-t1	frecord/test.tfre	cords
prepare-data-12_worker_1   2017-05-23T11:34:05.5742779742 bone running training code.	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.574277974	Z vone running tr	aining code.			
prepare-data-12_worker_1   2017-05-23T11:34:05.574318365Z Cannot find remote data volume , checkpoints are not persisted remotely.	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.57431836	5Z Cannot find rem	ote data volu	me , checkpoints	s are not persist	ed remotely.
prepare-data-12_worker_1   2017-05-23T11:34:05.574323498Z Done persisting checkpoints to remote storage.	prepare-d	lata–12_work	er_1	2017-05-2	23T11:34:	05.574323498	3Z Done persisting	checkpoints	to remote storag	je.	

E-MapReduce マシンにログインして、生成された TFRecord ファイルを確認することも で

きます。

```
dfs - ls / mnist - tfrecord
# hdfs
SLF4J : Class path contains multiple SLF4J
                                                  bindings .
SLF4J : Found
                binding in [ jar : file :/ opt / apps / hadoop
- 2 . 7 . 2 / share / hadoop / common / lib / slf4j - log4j12 - 1 .
7.10. jar ! / org / slf4j / impl / StaticLogg erBinder . class
SLF4J : Found binding in [ jar : file :/ opt / apps / tez - 0
. 8 . 4 / lib / slf4j - log4j12 - 1 . 7 . 10 . jar ! / org / slf4j
/ impl / StaticLogg erBinder . class ]
SLF4J : See http :// www . slf4j . org / codes . html #
multiple_b indings for an explanatio n .
SLF4J : Actual binding
                               of type [org.slf4j.impl.
                          is
Log4jLogge rFactory ]
Found 3
           items
```

- rw - r r	3 root	hadoop 8910000	2017 - 05 - 23	19
: 34 / mnist -	tfrecord /	test . tfrecords		
- rw - r r	3 root	hadoop 49005000	2017 - 05 - 23	19
: 33 / mnist -	tfrecord /	train . tfrecords		
- rw - r r	3 root	hadoop 4455000	2017 - 05 - 23	19
: 33 / mnist - tfrecord / validation . tfrecords				