Install Feast

Overview

This installation guide will demonstrate two ways of installing Feast:

- **Minikube (Minimal):** This installation has no external dependencies, but does not have a historical feature store installed. It allows users to quickly get a feel for Feast.
- ****Google Kubernetes Engine (Recommended):** _**_The guide that follows is a single cluster Feast installation on Google's GKE. It has Google Cloud specific dependencies like BigQuery, Dataflow, and Google Cloud Storage.

Minikube

Overview

This guide will install Feast into Minikube. Once Feast is installed you will be able to:

- Define and register features.
- Load feature data from both batch and streaming sources.
- Retrieve features for online serving.

{% hint style="warning" %} This Minikube installation guide is for demonstration purposes only. It is not meant for production use, and does not install a historical feature store. {% endhint %}

0. Requirements

The following software should be installed prior to starting:

1. Minikube should be installed.
2. Kubectl installed and configured to work with Minikube.
3. Helm (2.16.0 or greater).

1. Set up Minikube

Start Minikube. Note the minimum cpu and memory below:

```
minikube start --cpus=3 --memory=4096 --kubernetes-version='v1.15.5'
```

Set up your Feast environmental variables

```
export FEAST_IP=$(minikube ip)
export FEAST_CORE_URL=${FEAST_IP}:32090
export FEAST_SERVING_URL=${FEAST_IP}:32091
```

2. Install Feast with Helm

Clone the Feast repository and navigate to the `charts` sub-directory:

```
git clone https://github.com/gojek/feast.git && \
cd feast && export FEAST_HOME_DIR=$(pwd) && \
cd infra/charts/feast
```

Copy the `values-demo.yaml` file for your installation:

```
cp values-demo.yaml my-feast-values.yaml
```

Update all occurrences of the domain `feast.example.com` inside of `my-feast-values.yaml` with your Minikube IP. This is to allow external access to the services in the cluster. You can find your Minikube IP by running the following command `minikube ip` , or simply replace the text from the command line:

```
sed -i "s/feast.example.com/${FEAST_IP}/g" my-feast-values.yaml
```

Install Tiller:

```
helm init
```

Install the Feast Helm chart:

```
helm install --name feast -f my-feast-values.yaml .
```

Ensure that the system comes online. This will take a few minutes

```
watch kubectl get pods
```

```
NAME                                      READY   STATUS      RESTARTS
  AGE
pod/feast-feast-core-666fd46db4-l58l6     1/1     Running     0
   5m
pod/feast-feast-serving-online-84d99ddcbd 1/1     Running     0
   6m
pod/feast-kafka-0                         1/1     Running     0
   3m
pod/feast-kafka-1                         1/1     Running     0
   4m
pod/feast-kafka-2                         1/1     Running     0
   4m
pod/feast-postgresql-0                    1/1     Running     0
   5m
pod/feast-redis-master-0                  1/1     Running     0
   5m
pod/feast-zookeeper-0                     1/1     Running     0
   5m
pod/feast-zookeeper-1                     1/1     Running     0
   5m
pod/feast-zookeeper-2                     1/1     Running     0
   5m
```

3. Connect to Feast with the Python SDK

Install the Python SDK using pip:

```
pip install -e ${FEAST_HOME_DIR}/sdk/python
```

Configure the Feast Python SDK:

```
feast config set core_url ${FEAST_CORE_URL}
feast config set serving_url ${FEAST_SERVING_URL}
```

Make sure that both Feast Core and Feast Serving are connected:

```
feast version
```

```
{
  "sdk": {
    "version": "feast 0.3.2"
```

```
    },
    "core": {
      "url": "192.168.99.100:32090",
      "version": "0.3",
      "status": "connected"
    },
    "serving": {
      "url": "192.168.99.100:32091",
      "version": "0.3",
      "status": "connected"
    }
  }
```

That's it! You can now start to use Feast!

Google Kubernetes Engine

Overview

This guide will install Feast into a Kubernetes cluster on GCP. It assumes that all of your services will run within a single K8s cluster. Once Feast is installed you will be able to:

- Define and register features.
- Load feature data from both batch and streaming sources.
- Retrieve features for model training.
- Retrieve features for online serving.

{% hint style="info" %} This guide requires Google Cloud Platform for installation.

- BigQuery is used for storing historical features.
- Cloud Dataflow is used for running data ingestion jobs.
- Google Cloud Storage is used for intermediate data storage. {% endhint %}

0. Requirements

1. Google Cloud SDK installed, authenticated, and configured to the project you will use.
2. Kubectl installed.
3. Helm (2.16.0 or greater) installed on your local machine with Tiller installed in your cluster.

1. Set up GCP

First define the environmental variables that we will use throughout this installation. Please customize these to reflect your environment.

```
export FEAST_GCP_PROJECT_ID=my-gcp-project
export FEAST_GCP_REGION=us-central1
export FEAST_GCP_ZONE=us-central1-a
export FEAST_BIGQUERY_DATASET_ID=feast
export FEAST_GCS_BUCKET=${FEAST_GCP_PROJECT_ID}_feast_bucket
export FEAST_GKE_CLUSTER_NAME=feast
export FEAST_S_ACCOUNT_NAME=feast-sa
```

Create a Google Cloud Storage bucket for Feast to stage data during exports:

```
gsutil mb gs://${FEAST_GCS_BUCKET}
```

Create a BigQuery dataset for storing historical features:

```
bq mk ${FEAST_BIGQUERY_DATASET_ID}
```

Create the service account that Feast will run as:

```
gcloud iam service-accounts create ${FEAST_SERVICE_ACCOUNT_NAME}

gcloud projects add-iam-policy-binding ${FEAST_GCP_PROJECT_ID} \
  --member
serviceAccount:${FEAST_S_ACCOUNT_NAME}@${FEAST_GCP_PROJECT_ID}.iam.gservice
account.com \
  --role roles/editor

gcloud iam service-accounts keys create key.json --iam-account \
${FEAST_S_ACCOUNT_NAME}@${FEAST_GCP_PROJECT_ID}.iam.gserviceaccount.com
```

Ensure that Dataflow API is enabled:

```
gcloud services enable dataflow.googleapis.com
```

2. Set up a Kubernetes (GKE) cluster

{% hint style="warning" %} Provisioning a GKE cluster can expose your services publicly. This guide does not cover securing access to the cluster. {% endhint %}

Create a GKE cluster:

```
gcloud container clusters create ${FEAST_GKE_CLUSTER_NAME} \
    --machine-type n1-standard-4
```

Create a secret in the GKE cluster based on your local key `key.json`:

```
kubectl create secret generic feast-gcp-service-account --from-
file=key.json
```

For this guide we will use `NodePort` for exposing Feast services. In order to do so, we must find an internal IP of at least one GKE node.

```
export FEAST_IP=$(kubectl describe nodes | grep InternalIP | awk '{print
$2}' | head -n 1)
export FEAST_CORE_URL=${FEAST_IP}:32090
export FEAST_ONLINE_SERVING_URL=${FEAST_IP}:32091
export FEAST_BATCH_SERVING_URL=${FEAST_IP}:32092
```

Confirm that you are able to access this node:

```
ping $FEAST_IP
```

```
PING 10.123.114.11 (10.203.164.22) 56(84) bytes of data.
64 bytes from 10.123.114.11: icmp_seq=1 ttl=63 time=54.2 ms
64 bytes from 10.123.114.11: icmp_seq=2 ttl=63 time=51.2 ms
```

3. Set up Helm

Run the following command to provide Tiller with authorization to install Feast:

```
kubectl apply -f - <<EOF
apiVersion: v1
kind: ServiceAccount
metadata:
  name: tiller
  namespace: kube-system
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: tiller
```

```
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: cluster-admin
  subjects:
    - kind: ServiceAccount
      name: tiller
      namespace: kube-system
  EOF
```

Install Tiller:

```
  helm init --service-account tiller
```

4. Install Feast with Helm

Clone the Feast repository and navigate to the `charts` sub-directory:

```
  git clone https://github.com/gojek/feast.git && cd feast && \
  git checkout 0.3-dev && \
  export FEAST_HOME_DIR=$(pwd) && \
  cd infra/charts/feast
```

Make a copy of the Helm `values.yaml` so that it can be customized for your Feast deployment:

```
  cp values.yaml my-feast-values.yaml
```

Update `my-feast-values.yaml` based on your GCP and GKE environment.

- Required fields are paired with comments which indicate whether they need to be replaced.
- All occurrences of `feast.example.com` should be replaced with either your domain name or the IP stored in `$FEAST_IP`.

Install the Feast Helm chart:

```
  helm install --name feast -f my-feast-values.yaml .
```

Ensure that the system comes online. This will take a few minutes

```
  watch kubectl get pods
```

```
NAME                                      READY   STATUS      RESTARTS
  AGE
pod/feast-feast-core-666fd46db4-l58l6     1/1     Running     0
   5m
pod/feast-feast-serving-online-84d99ddcbd 1/1     Running     0
   6m
pod/feast-kafka-0                         1/1     Running     0
   3m
pod/feast-kafka-1                         1/1     Running     0
   4m
pod/feast-kafka-2                         1/1     Running     0
   4m
pod/feast-postgresql-0                    1/1     Running     0
   5m
pod/feast-redis-master-0                  1/1     Running     0
   5m
pod/feast-zookeeper-0                     1/1     Running     0
   5m
pod/feast-zookeeper-1                     1/1     Running     0
   5m
pod/feast-zookeeper-2                     1/1     Running     0
   5m
```

5. Connect to Feast with the Python SDK

Install the Python SDK using pip:

```
pip install -e ${FEAST_HOME_DIR}/sdk/python
```

Configure the Feast Python SDK:

```
feast config set core_url ${FEAST_CORE_URL}
feast config set serving_url ${FEAST_ONLINE_SERVING_URL}
```

Make sure that both Feast Core and Feast Serving are connected:

```
feast version
```

```
{
  "sdk": {
    "version": "feast 0.3.2"
```

```
    },
    "core": {
      "url": "192.168.99.100:32090",
      "version": "0.3",
      "status": "connected"
    },
    "serving": {
      "url": "192.168.99.100:32091",
      "version": "0.3",
      "status": "connected"
    }
  }
```

That's it! You can now start to use Feast!