* Anthos is an open-source application platform that enables you to modernize your existing applications in your hybrid or multi-cloud environment.
* You can build new VMs and run them anywhere in a secure manner. Anthos is built on open source technologies pioneered by Google —
* including Kubernetes, Istio, and Knative — and enables consistency between on-premises and cloud environments.

**Create the source Compute Engine:**

gcloud compute instances create source-vm --zone=us-west4-b --machine-type=e2-standard-2 --subnet=default --scopes="cloud-platform" --tags=http-server,https-server --image=ubuntu-minimal-1604-xenial-v20210119a --image-project=ubuntu-os-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard --boot-disk-device-name=source-vm \

--metadata startup-script=METADATA\_SCRIPT

A black screen with white text

Description automatically generated

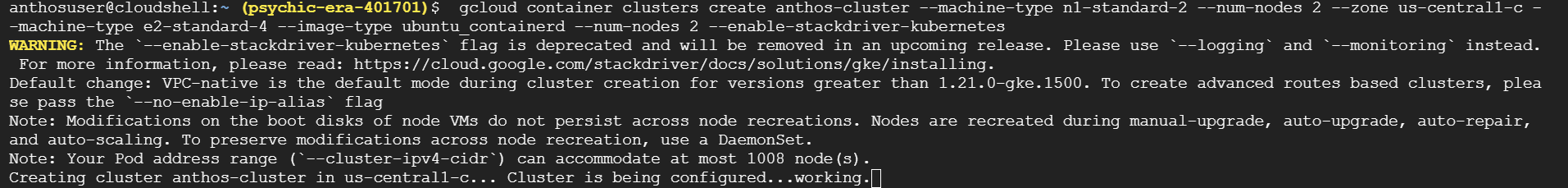
**To migrate VM to a container first stop the VM:**

A screenshot of a computer

Description automatically generated

**Create a processing cluster**:

gcloud container clusters create anthos-cluster --machine-type n1-standard-2 --num-nodes 2 --zone us-central1-c --machine-type e2-standard-4 --image-type ubuntu\_containerd --num-nodes 2 --enable-stackdriver-kubernetes



A black and white screen

Description automatically generated

**Install Migrate to Containers**:

In Cloud Shell create the m4a-install service account:

gcloud iam service-accounts create m4a-install



Grant the storage.admin role to the service account:

gcloud projects add-iam-policy-binding psychic-era-401701 --member="serviceAccount:m4a-install@psychic-era-401701.iam.gserviceaccount.com" --role="roles/storage.admin"

A black screen with white text

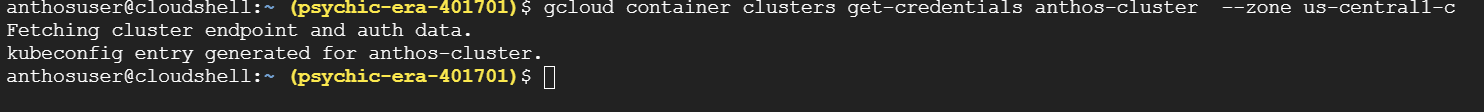
Description automatically generated

Download the key file for the service account:

gcloud iam service-accounts keys create m4a-install.json --iam-account=m4a-install@psychic-era-401701.iam.gserviceaccount.com --project=psychic-era-401701

**Connect to the cluster:**

gcloud container clusters get-credentials anthos-cluster --zone us-central1-c



Set up Migrate to Containers components on your processing cluster by using the migctl command-line tool included with Migrate to Containers:

migctl setup install --json-key=m4a-install.json --gcp-project psychic-era-401701 --gcp-region us-central1

A screenshot of a computer program

Description automatically generated

Validate the Migrate to Containers installation. Use the migctl doctor command to confirm a successful deployment:

A screen shot of a computer

Description automatically generated

Re-run the command until you see the successful deployment.

A screenshot of a computer

Description automatically generated

**Migrating the VM:**

Now you'll create a migration plan with migration details, then use it to migrate the VM.

To use Compute Engine as a migration source, you must first create a service account with the compute.viewer and compute.storageAdmin roles:

**In Cloud Shell create the m4a-ce-src service account:**

gcloud iam service-accounts create m4a-ce-src --project=psychic-era-401701

Grant the compute.viewer role to the service account:

gcloud projects add-iam-policy-binding psychic-era-401701 \

--member="serviceAccount:m4a-ce-src@psychic-era-401701.iam.gserviceaccount.com" \

--role="roles/compute.viewer"

**Grant the compute.storageAdmin role to the service account:**

gcloud projects add-iam-policy-binding psychic-era-401701 \

--member="serviceAccount:m4a-ce-src@psychic-era-401701.iam.gserviceaccount.com" \

--role="roles/compute.storageAdmin"

**Download the key file for the service account:**

gcloud iam service-accounts keys create m4a-ce-src.json \

--iam-account=m4a-ce-src@psychic-era-401701.iam.gserviceaccount.com \

--project=psychic-era-401701

Create the migration source:

migctl source create ce source-vm --project psychic-era-401701 --json-key=m4a-ce-src.json

A screen shot of a computer

Description automatically generated

Where m4a-ce-src.json specifies the service account you created above.

**Create a migration:**

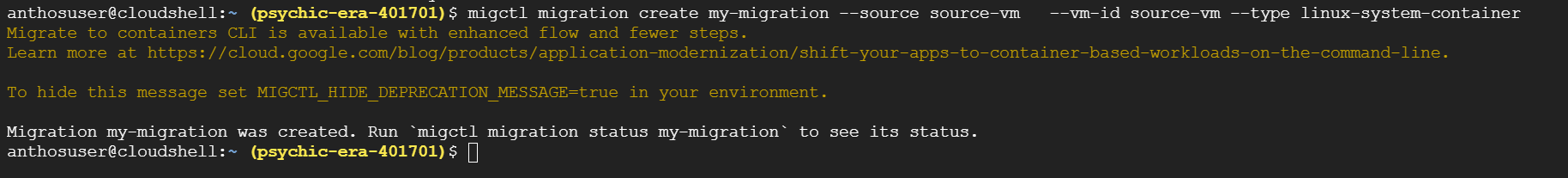
You begin migrating VMs by creating a migration. This results in a migration plan object.

Migration is the central object with which you perform migration actions, monitor migration activities and status with the migctl tool or in the Cloud Console. The migration object is implemented as a Kubernetes Custom Resource Definition (CRD).

Next you will create a migration by running the migctl tool.

**Create the migration plan that defines what to migrate:**

migctl migration create my-migration --source source-vm --vm-id source-vm --type linux-system-container



Run the following to check the status

migctl migration status my-migration

A black screen with white text

Description automatically generated

**Review the migration plan:**

migctl migration get my-migration

A screenshot of a computer

Description automatically generated

**Migrate the VM using the migration plan:**

migctl migration generate-artifacts my-migration

A screen shot of a computer

Description automatically generated

migctl migration status my-migration:

A screen shot of a computer

Description automatically generated

The migration can take several minutes

Re-run the command until you see the Completed status.

A black screen with white text

Description automatically generated

You can add -v flag for verbosity:

migctl migration status my-migration -v

A black screen with white text

Description automatically generated

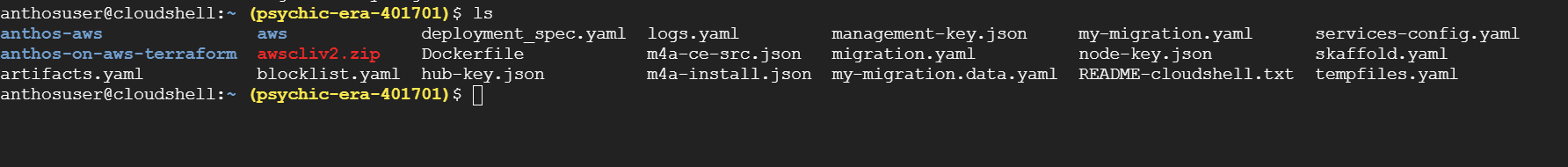
In the following steps you'll get the deployment artifacts you generated during migration,

then use them to deploy your migrated workload to the cluster.

migctl migration get-artifacts my-migration

A black screen with white text

Description automatically generated



deployment\_spec.yaml -- The YAML file that configures your workload.

Dockerfile -- The Dockerfile used to build the image for your migrated VM.

migration.yaml -- A copy of the migration plan.

blocklist.yaml -- The list of container services to disable based on your settings in the migration plan.

If the Cloud Shell editor isn't open already, click the Open Editor button.

Open the deployment\_spec.yaml file and locate the Service object whose name is source-vm.

Beneath the following Service definition, add another Service at the end that will expose port 80 for access to your web server over HTTP:

apiVersion: v1

kind: Service

metadata:

name: my-service

spec:

selector:

app: source-vm

ports:

- protocol: TCP

port: 80

targetPort: 80

type: LoadBalancer

A screen shot of a computer

Description automatically generated

Save the file.

Apply the deployment\_spec.yaml to deploy the workload:

kubectl apply -f deployment\_spec.yaml

A black screen with white text

Description automatically generated

Now check for an external IP address:

kubectl get service

A screen shot of a computer

Description automatically generated

**Test the migration**

Test the migration by opening a browser and visiting the web page at the external IP address of my-service (be sure to use HTTP rather than HTTPS).

For example:

http://34.136.89.32