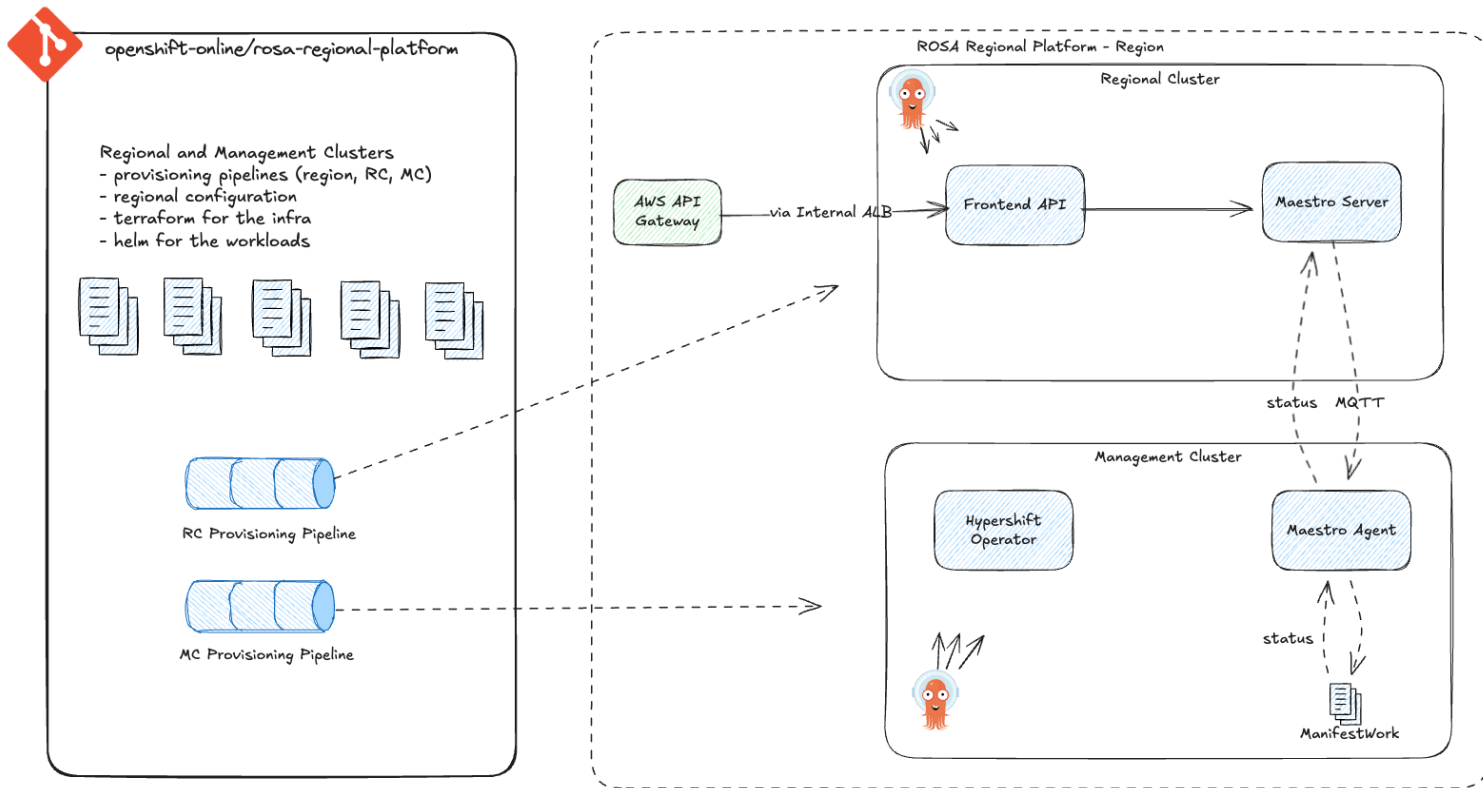


ROSA Regional Platform

✓ ROSA-666 - Milestone 1 - Deploy a Regional Cluster + Management Cluster (part of ROSA-659).

A platform engineer can deploy a new region manually, composed of an EKS Regional Cluster (RC) and EKS Management Cluster (MC). They can submit a resource via the Frontend API, and see it applied in the MC via Maestro, and receive status feedback.

What We've Built



Complete Guide: Provision a New Region

docs / full-region-provisioning.md

The screenshot shows a GitHub repository interface for 'rosal-region-platform'. The file 'docs / full-region-provisioning.md' is selected, showing 368 lines of code. The document content includes:

Complete Guide: Provision a New Region

This comprehensive guide walks through all steps to provision a new region in the ROSA Regional Platform. Follow these steps in order to set up both Regional and Management Clusters with full ArgoCD configuration and Maestro connectivity.

1. Pre-Flight Checklist

Before starting, ensure your environment is properly configured.

Required Tools

Verify all tools are installed and accessible:

```
# Check tool versions
aws --version
terraform --version
python --version # or python3 --version
```

Required AWS accounts

To provision a regional and management cluster, you require two AWS accounts. Ensure you have access to both via environment variables or ideally AWS profiles.

Declare a New Region

Define your region in Git, render the ArgoCD manifests, commit.

1. Edit argocd/config.yaml

```
shards:
  - region: "us-west-2"
    environment: "integration"
    values:
      management-cluster:
        hypershift:
          oidcStorageS3Bucket:
            name: "hypershift-mc-us-west-2"
            region: "us-west-2"
```

2. Render & Commit

```
# Generate all the argocd/rendered
# manifests for the new region based
# on the config.yaml values /argocd/scripts/render.py

git add argocd/config.yaml argocd/rendered/
git commit -m "Add us-west-2 region"
git push
```

Regional Cluster Provisioning

Provision the Regional Cluster infrastructure with Terraform.

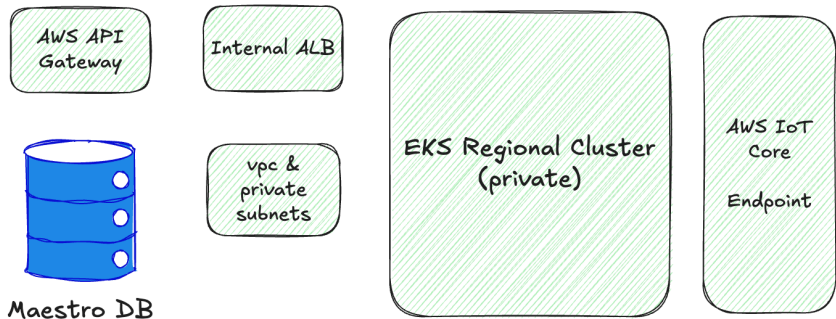
Commands

```
# Configure terraform variables
cp terraform/.../terraform.tfvars.example \
  terraform/.../terraform.tfvars

# Provision Regional Cluster
make provision-regional
```

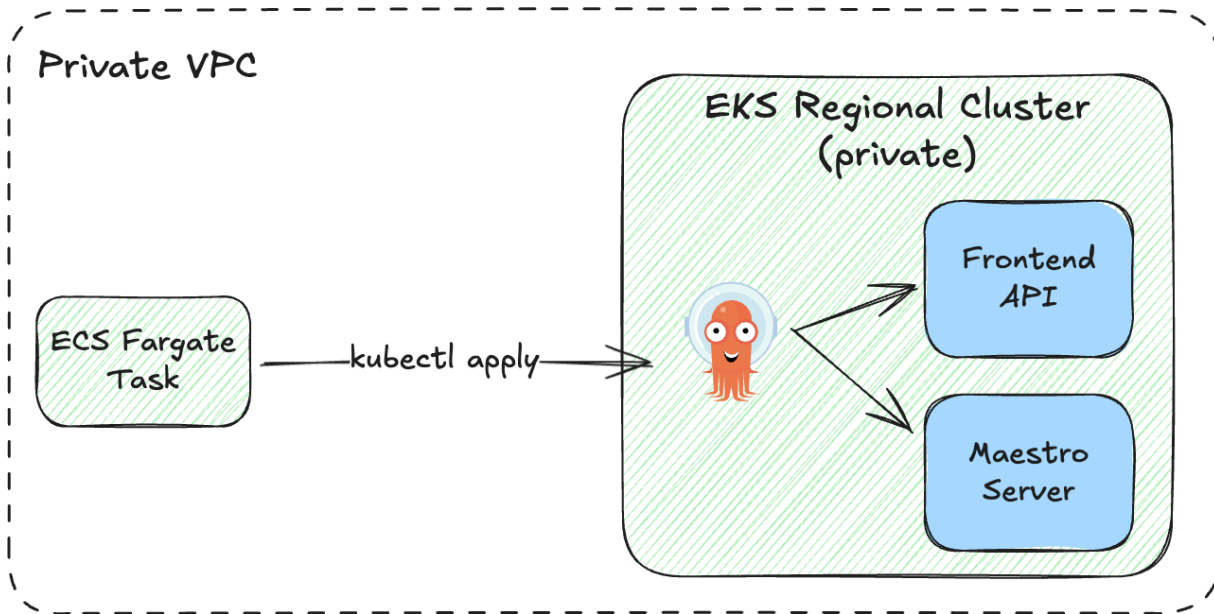
This runs a terraform that provisions all the Regional Cluster infrastructure.

What Gets Created



ArgoCD Bootstrap & Workloads

As part of `make provision-regional`, an **ECS Fargate task** runs to **install ArgoCD**, then GitOps takes over. **ArgoCD** automatically **deploys all Regional Cluster workloads** from Git.



Maestro IoT Setup

All the clusters (Regional and Management Cluster) communicate using Maestro over MQTT. This requires:

```
# In the Regional Account - Provision IoT resources  
make provision-maestro-agent-iot-regional
```

This creates:

- Certificate + Private Key (download files)
- Set up IoT Policy for that certificate

```
# Then, in the Management Account  
make provision-maestro-agent-iot-management
```

This uploads the certificate to AWS Secrets Manager and will be referenced by the Maestro Agent deployment.

Management Cluster Provisioning

Provision the MC and configure Maestro Agent with IoT credentials.

Commands

```
# Provision Management Cluster  
make provision-management
```

This runs a terraform that provisions the Management Cluster.

What Gets Created

- Private EKS Cluster
- ArgoCD (self-managing)
- HyperShift Operator (deployed by ArgoCD)
- Maestro Agent (deployed by ArgoCD) configured with IoT credentials

Consumer Registration

Register the MC with the Regional Cluster's Maestro Server.

```
awscurl -X POST \  
https://$API_GATEWAY_URL/prod/api/v0/management_clusters \  
--service execute-api \  
--region us-west-2 \  
-H "Content-Type: application/json" \  
-d '{  
  "name": "management-01",  
  "labels": {  
    "cluster_type": "management",  
    "cluster_id": "management-01"  
  }  
'
```

What This Does

Registers MC as a consumer in Maestro.

The `API_GATEWAY_URL` can be found in the Regional Cluster terraform output.

End-to-End Verification

Apply a workload through the API and verify it reaches the MC.

Apply Workload

```
# Post a ManifestWork
awscurl -X POST \
  https://$API_GATEWAY_URL/prod/api/v0/work \
  --service execute-api \
  --region us-west-2 \
  -d @payload.json

# Check status
awscurl \
  https://$API_GATEWAY_URL/prod/api/v0/resource_bundles \
  --service execute-api \
  --region us-west-2 | jq '.items[].status'
```

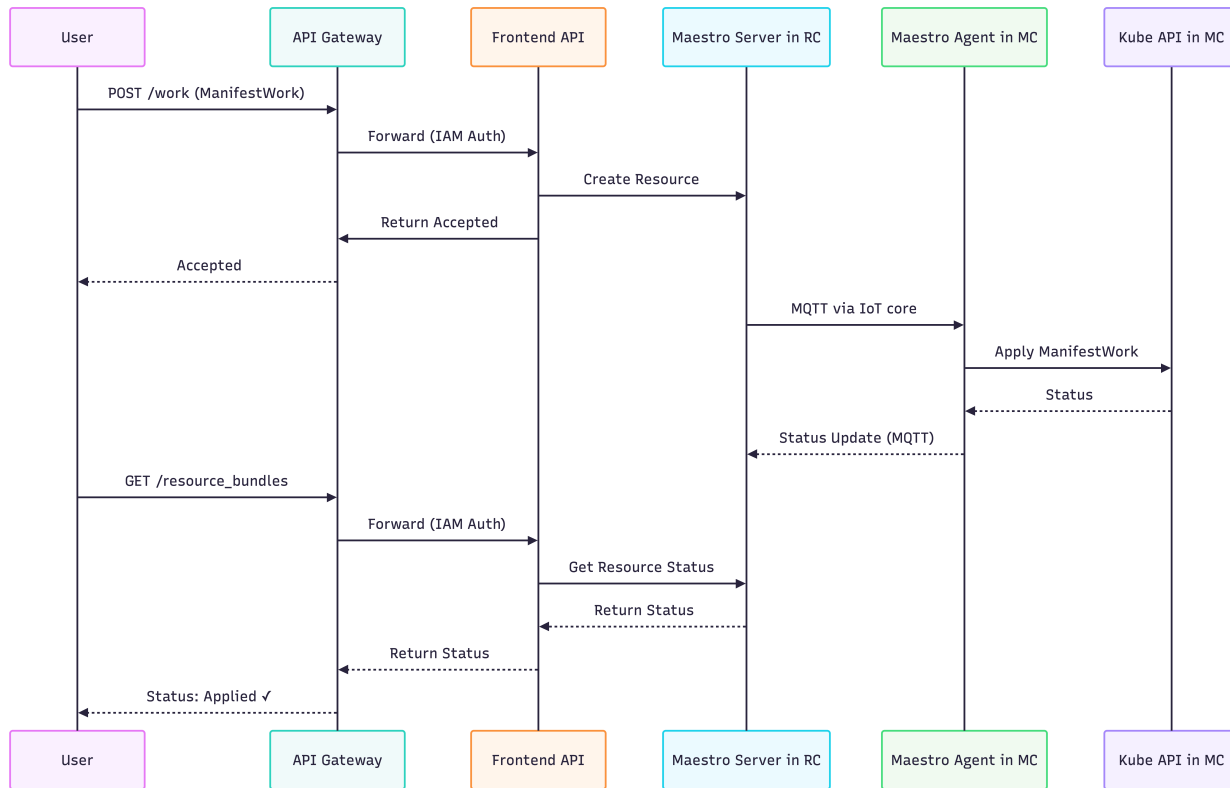
The `payload.json` is a ManifestWork resource documented in the End-to-End Verification guide.

Expected Result

```
{
  "resourceStatus": {
    "conditions": [{
      "type": "Applied",
      "status": "True"
    }]
  }
}
```

✅ Workload successfully distributed via Maestro

Request Flow



Milestone 1 Complete

Regional Cluster and Management Cluster provisioned on EKS.

What's next?

ROSA-667 - Milestone 2 - Continuous Validation

ROSA-668 - Milestone 3 - HCPs run on EKS MCs