Deploy an AWS IaaS Cluster with Crossplane

Palette supports using Crossplane to create and manage Kubernetes host clusters across major infrastructure providers. This section guides you on how to use Crossplane to deploy a Palette-managed Kubernetes cluster in AWS.

Prerequisites

- A Palette account and API key. Refer to the Create API Key page for instructions on creating an API key.
- A public AWS cloud account with the required IAM Policies.
- An SSH key pair available in the region where you want to deploy the cluster. Check out the AWS Create EC2 SSH Key Pair page for guidance.
- The AWS account must be registered in Palette. Follow the Add an AWS Account to Palette guide to register your account in Palette.
- A Kubernetes cluster with at least 2 GB of RAM. This guide uses a kind cluster as an example. Refer to the kind Quick Start guide to learn how to install kind and create a cluster.
- The following software must be installed on your host:
  - curl
  - jq
  - Helm version v3.2.0 or later
  - kubectl
  - A text editor such as Vi or Nano. This guide uses Vi as an example.

Feedback

Ask AI

Deploy an AWS IaaS Cluster with Crossplane

1. Open a terminal session and set the kubectl context to your cluster. Replace `<cluster-name>` with the name of your cluster.

```
kubectl cluster-info --context <cluster-name>
```

Example output

```
Kubernetes control plane is running at https://127.0.0.1:65306
CoreDNS is running at https://127.0.0.1:65306/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

2. Next, add the Crossplane Helm chart.

```
helm repo add \
crossplane-stable https://charts.crossplane.io/stable
helm repo update
```

Example output

```
"crossplane-stable" has been added to your repositories
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "ingress-nginx" chart repository
...Successfully got an update from the "crossplane-stable" chart repository
Update Complete. ⎈Happy Helming!⎈
```

3. Install the Crossplane components using the `helm install` command.

```
helm install crossplane \
crossplane-stable/crossplane \
```

```
--namespace crossplane-system \
--create-namespace
```

Example output

```
NAME: crossplane
LAST DEPLOYED: Fri Aug  8 08:27:46 2025
NAMESPACE: crossplane-system
STATUS: deployed
REVISION: 1
TEST SUITE: None
NOTES:
Release: crossplane

Chart Name: crossplane
Chart Description: Crossplane is an open source Kubernetes add-on
that enables platform teams to assemble infrastructure from multiple
vendors, and expose higher level self-service APIs for application
teams to consume.
Chart Version: 1.20.0
Chart Application Version: 1.20.0

Kube Version: v1.33.1
```

Verify the installation with the `kubectl get pods` command. The output must contain two Crossplane pods with a `Running` status.

```
kubectl get pods --namespace crossplane-system
```

Example output

```
NAME                                       READY   STATUS    RESTARTS
AGE
crossplane-869d89c8f8-7jc6c                1/1     Running   0
20s
crossplane-rbac-manager-784b496b-8mr6z     1/1     Running   0
20s
```

4. Once Crossplane is installed, create a folder to store the Kubernetes configuration files.

```
mkdir crossplane-aws
```

5. Fetch the latest version of the Palette Crossplane provider. Alternatively, identify your desired version from the Upbound Marketplace and set the value manually.

```
PALETTE_CROSSPLANE_PROVIDER_VERSION=$(curl --silent
https://api.github.com/repos/crossplane-contrib/provider-
palette/releases/latest | jq --raw-output .tag_name)
echo Palette Crossplane Provider Version:
$PALETTE_CROSSPLANE_PROVIDER_VERSION
```

6. Create the following Kubernetes configuration for the Palette Crossplane provider.

```
cat << EOF > crossplane-aws/provider-palette.yaml
apiVersion: pkg.crossplane.io/v1
kind: Provider
metadata:
  name: provider-palette
spec:
  package: xpkg.upbound.io/crossplane-contrib/provider-
palette:$PALETTE_CROSSPLANE_PROVIDER_VERSION
EOF
```

Verify that the file was created and populated with the expected Palette Crossplane provider version.

```
cat crossplane-aws/provider-palette.yaml
```

Example output

```
apiVersion: pkg.crossplane.io/v1
kind: Provider
```

```
metadata:
  name: provider-palette
spec:
  package: xpkg.upbound.io/crossplane-contrib/provider-
palette:v0.24.0
```

7. Issue the command below to install the Palette Crossplane provider. Crossplane installs the CRDs that allow you to create Palette resources directly inside Kubernetes.

```
kubectl apply --filename crossplane-aws/provider-palette.yaml
```

Example output

```
provider.pkg.crossplane.io/provider-palette created
```

Check the installation with the `kubectl get providers` command.

```
kubectl get providers
```

Example output

```
NAME               INSTALLED   HEALTHY   PACKAGE
AGE
provider-palette   True        True      xpkg.upbound.io/crossplane-
contrib/provider-palette:v0.24.0   40s
```

8. Set the following variables for your Palette environment:

   - `<palette-api-key>` - Your Palette API key.
   - `<palette-project-name>` - The name of the Palette project you are deploying your cluster in.
   - `<palette-endpoint>` - The endpoint of your Palette environment.

```
PALETTE_API_KEY=<palette-api-key>
PALETTE_PROJECT_NAME=<palette-project-name>
PALETTE_HOST=<palette-endpoint>
```

9. Create a file to store the Kubernetes Secret containing your Palette API key and environment details. The Palette provider requires credentials to create and manage resources.

```
cat << EOF > crossplane-aws/secret-aws.yaml
apiVersion: v1
kind: Secret
metadata:
  name: palette-creds
  namespace: crossplane-system
type: Opaque
stringData:
  credentials: |
    {
    "api_key": "$PALETTE_API_KEY",
    "project_name": "$PALETTE_PROJECT_NAME",
    "host": "$PALETTE_HOST"
    }
EOF
```

Verify that the file was created and populated with the expected API key and environment values.

```
cat crossplane-aws/secret-aws.yaml
```

Example output

```
apiVersion: v1
kind: Secret
metadata:
  name: palette-creds
  namespace: crossplane-system
type: Opaque
```

```
stringData:
  credentials: |
    {
    "api_key": "**************",
    "project_name": "Default",
    "host": "console.spectrocloud.com"
    }
```

10. Create the Kubernetes Secret.

```
kubectl apply --filename crossplane-aws/secret-aws.yaml
```

Example output

```
secret/palette-creds created
```

11. Create a file to store the `ProviderConfig` object. This object configures the Palette Crossplane provider with the Secret containing the Palette API key.

```
cat << EOF > crossplane-aws/providerconfig-aws.yaml
apiVersion: palette.crossplane.io/v1beta1
kind: ProviderConfig
metadata:
  name: default
spec:
  credentials:
    source: Secret
    secretRef:
      name: palette-creds
      namespace: crossplane-system
      key: credentials
EOF
```

12. Create the Kubernetes `ProviderConfig` object.

```
kubectl apply --filename crossplane-aws/providerconfig-aws.yaml
```

Example output

```
providerconfig.palette.crossplane.io/default created
```

13. Once the Palette Crossplane provider is installed and set up, create a file to store the cluster profile configuration.

> **WARNING**
>
> We recommend creating the cluster profile file first and directly pasting the contents into the file. Redirecting cluster profile configurations from the terminal into the file can misinterpret escape characters, resulting in an invalid file.

```
vi crossplane-aws/cluster-profile-aws.yaml
```

14. Paste the Kubernetes configuration below into the text editor window that opens. Save the file and exit.

```
apiVersion: cluster.palette.crossplane.io/v1alpha1
kind: Profile
metadata:
  name: aws-crossplane-cluster-profile
  namespace: crossplane-system
spec:
  forProvider:
    cloud: "aws"
    description: "AWS Crossplane cluster profile"
    name: "aws-crossplane-cluster-profile"
    type: "cluster"
    pack:
      - name: "ubuntu-aws"
        tag: "22.04"
        uid: "63bd0373764141c6622c3062"
        registryUid: "5eecc89d0b150045ae661cef"
        values:
          "# Spectro Golden images includes most of the hardening as pe
          standards\n\n# Uncomment below section to\n# 1. Include custo
```

The cluster profile contains the following core infrastructure layers.

Pack Type	Registry	Pack Name	Pack Version
OS	Public Repo	`ubuntu-aws`	`22.04`
Kubernetes	Public Repo	`kubernetes`	`1.32.4`
Network	Public Repo	`cni-calico`	`3.30.1`
Storage	Public Repo	`csi-aws-ebs`	`1.43.0`

> TIP
>
> If you want to use different packs in your cluster profile, use the Palette UI to simulate creating a cluster profile to gather the pack's required values. During the cluster profile creation, select the **API** button in the top-right to display the API payload. Replace the values of each pack's `name`, `tag`, `uid`, `registryUid`, and `values` as necessary. For information on creating cluster profiles, refer to our Create Cluster Profiles guide.

15. Create the cluster profile in Palette.

```
kubectl apply --filename crossplane-aws/cluster-profile-aws.yaml
```

Example output

```
profile.cluster.palette.crossplane.io/aws-crossplane-cluster-profile
created
```

16. Issue the commands below to get the ID of the cluster profile once it is created and save it as a variable.

```
kubectl wait --for=condition=Ready
profile.cluster.palette.crossplane.io/aws-crossplane-cluster-profile
```

```
CLUSTER_PROFILE_ID=$(kubectl get
profile.cluster.palette.crossplane.io aws-crossplane-cluster-profile
--output jsonpath='{.status.atProvider.id}')
echo Cluster Profile ID: $CLUSTER_PROFILE_ID
```

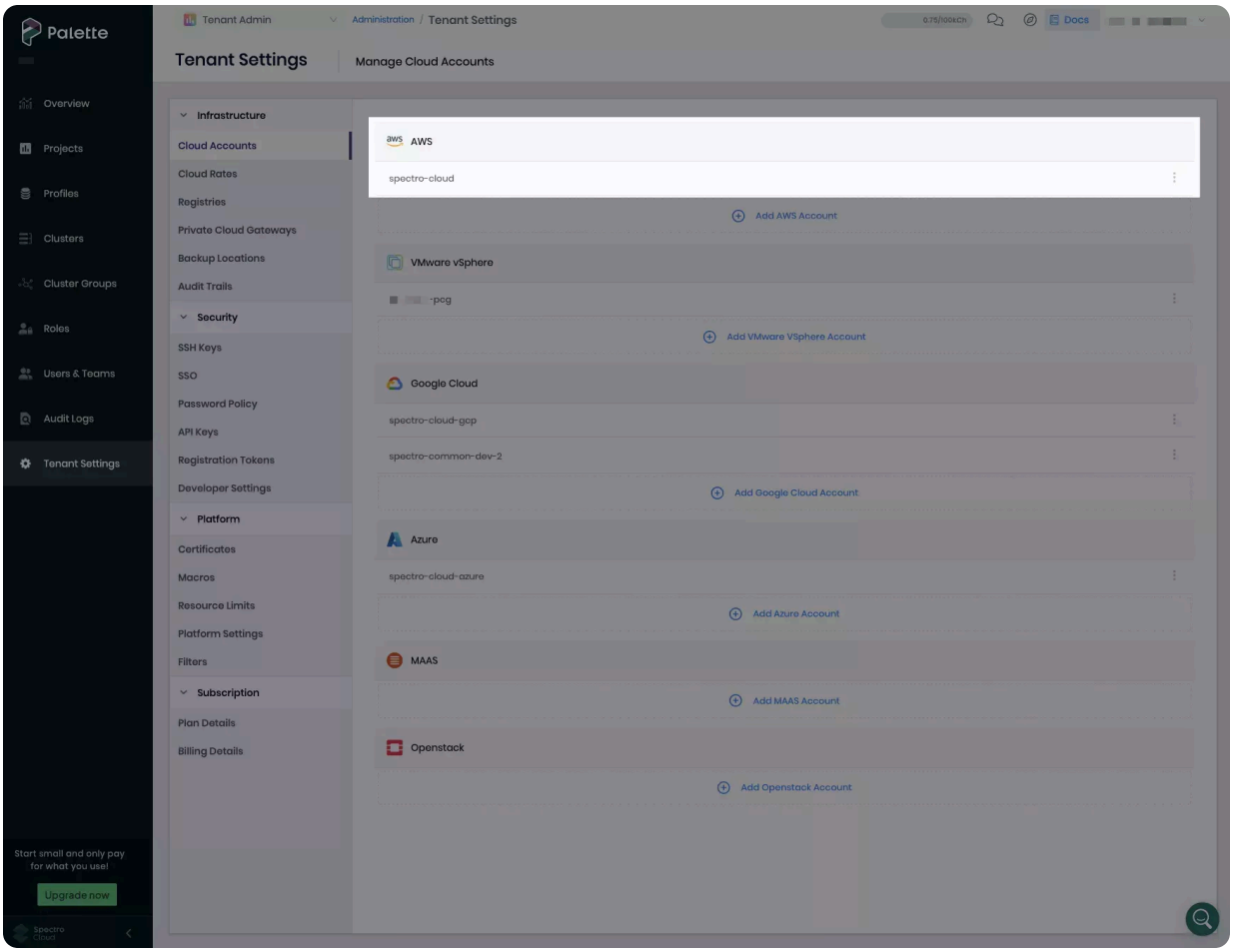
Example output

```
profile.cluster.palette.crossplane.io/aws-crossplane-cluster-profile
condition met
Cluster Profile ID: 68960ddf222fa7f0046e80ed
```

17. Next, set your Palette AWS account name as a variable. Replace `<aws-account-name>` with the name under which you registered your AWS account in Palette. This is the display name that appears under **Cloud Accounts** in **Tenant Settings** or **Project Settings**, *not* the actual name of your AWS account.

```
PALETTE_AWS_CLOUD_ACCOUNT_NAME=<aws-account-name>
```

In the example below, the Palette AWS account name is `spectro-cloud`.



18. Next, fetch the ID of your AWS cloud account registered in Palette by invoking the `cloudaccounts` Palette API.

```
AWS_CLOUD_ACCOUNT_ID=$(curl --location --request GET
'https://api.spectrocloud.com/v1/cloudaccounts/aws' \
-H 'Accept: application/json' \
-H "ApiKey: $PALETTE_API_KEY" \
| jq --arg name "$PALETTE_AWS_CLOUD_ACCOUNT_NAME" '.items[] |
select(.metadata.name == $name) | .metadata.uid' -r)
echo Cloud Account ID: $AWS_CLOUD_ACCOUNT_ID
```

19. Use the following command to create a file to store your AWS IaaS cluster configuration. Replace `<ssh-key-name>` with the name of your AWS EC2 SSH key that belongs to the region where you want to deploy your cluster.

    Optionally, edit the region, availability zone, instance type, and number of cluster nodes according to your workload.

```
cat << EOF > crossplane-aws/cluster-aws.yaml
apiVersion: cluster.palette.crossplane.io/v1alpha1
kind: Aws
metadata:
  name: aws-crossplane-cluster
  namespace: crossplane-system
spec:
  forProvider:
    name: aws-crossplane-cluster
    cloudConfig:
      - sshKeyName: <ssh-key-name>
        region: us-east-1
    machinePool:
      - azs:
          - us-east-1a
        count: 2
        instanceType: t3.xlarge
        name: machinepool1
      - azs:
```

20. Create the AWS IaaS cluster.

```
kubectl apply --filename crossplane-aws/cluster-aws.yaml
```

Example output

```
aws.cluster.palette.crossplane.io/aws-crossplane-cluster created
```

21. Wait for the cluster to be created. Cluster provisioning may take up to 20 minutes.

```
kubectl wait --for=condition=Ready
aws.cluster.palette.crossplane.io/aws-crossplane-cluster --timeout=1h
```

Once ready, you should receive output similar to the following.

Example output

```
aws.cluster.palette.crossplane.io/aws-crossplane-cluster condition
met
```

Validate

1. Log in to Palette.
2. From the left main menu, select **Clusters**.
3. Verify the deployed cluster named `aws-crossplane-cluster` is displayed and has a **Cluster Status** of **Running** and a **Health** status of **Healthy**.

Tags: crossplane aws iac infrastructure as code

*Last updated on **Aug 12, 2025***