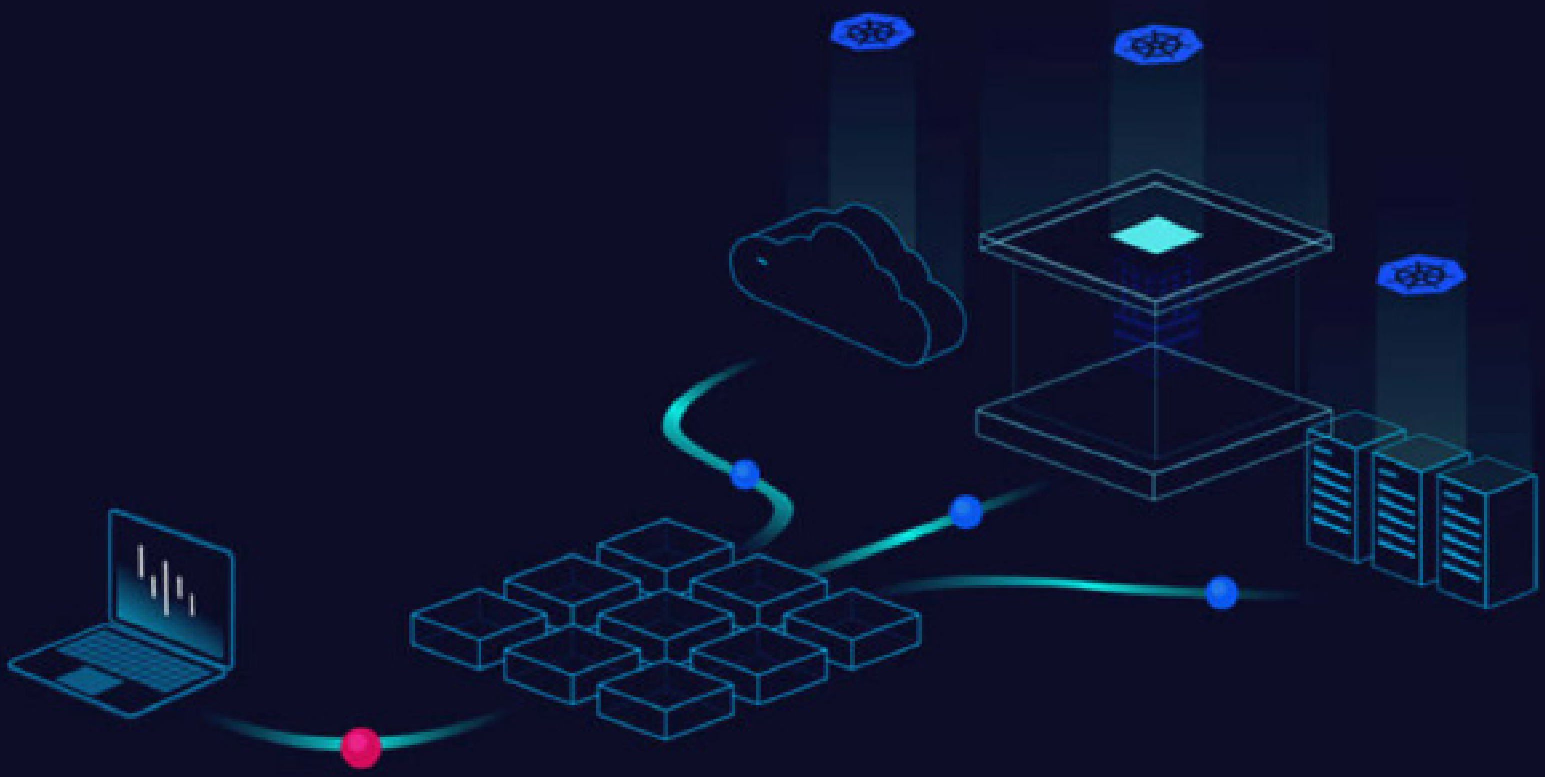
PLATFORM9

White Paper

Managing Kubernetes: Kops vs. AWS EKS vs. Platform9 KaaS



Contents

Overview

Software engineering organizations can consider a number of different options when evaluating which Kubernetes management solution is right for them. This article provides DevOps engineers with a clear picture of the similarities and differences between three popular Kubernetes deployment and management tools: Kubernetes Operations (Kops), AWS Elastic Kubernetes Service (EKS), and Platform9 KaaS.

This article assumes that the reader is already familiar with Kubernetes; if not, you should review the Kubernetes documentation for a very clear explanation of what you need to know to get started. Platform9 has also posted a number YouTube videos covering all aspects of setting up and managing Kubernetes.

IT teams often want to move from Kops to Platform9 because they have found Kops to be difficult to set up and manage. Some teams first try Amazon EKS as their Kubernetes solution, but want to replace it with a multicloud solution that doesn't have the limitations of EKS.

IT teams often want to move from Kops to Platform9 because they have found Kops to be difficult to set up and manage. Some teams first try Amazon EKS as their Kubernetes solution, but want to replace it with a multicloud solution that doesn't have the limitations of EKS.

To show you how Kops, EKS, and Platform9 compare, we will examine each option based on the following:

- Setup and deployment
- Configuration and access
- Cluster management
- Portability, multicloud support, and lock-In
- Networking and load balancing

At-a-Glance

The following table compares the ease-of-implementation and key features of the three solutions.

	Kops	**EKS**	**Platform9**
Setup and Deployment	Moderate (CLI only), requires some CLI knowledge to complete.	Simple to complex, depending on your chosen deployment methodology such as CLI vs. UI.	Simple setup and deployment using web UI. CLI setup is also available.
Configuration and Access	Complex (CLI only), requires some CLI knowledge to complete.	Moderate, requires knowledge of other AWS services such as IAM and ECR.	Simple configuration for cluster via web UI. User access creation available via web UI as well. CLI also available.
Cluster Management	Complex (CLI only), requires more advanced CLI knowledge along with other tools such as kubectl.	Moderate, control plane is managed by AWS. Scaling and container deployment is user managed.	Simple to manage. Control plane is abstracted and managed by Platform9. Scaling easily done via CLI or web UI.
Portability	Good, but doesn't run on bare metal or VMs outside of supported cloud providers.	Everything works best with AWS-based solutions (except containers which are portable).	Excellent portability across cloud providers as well as bare metal.
Networking	Good, but requires knowledge of CNI and Kubernetes network plugins.	Good when using AWS-based solutions.	Excellent. Multiple network plugins available depending on user requirements.

Kops

Kops is the Kubernetes community's officially supported method of setting up and maintaining Kubernetes clusters in the cloud. It is a CLI-based tool and is usually installed alongside kubectl for complete cluster management. It offers a number of features that are worth considering when you're evaluating which solution you want to use for cluster setup.

Since Kops is supported directly by the Kubernetes community, it's often the first solution that teams try. The most common issue that teams run into with Kops (depending on the use case), is the complexity of managing it.

Features

- Granular control of Kubernetes cluster setup in the cloud via a CLI-based tool that can be installed locally on your laptop or a virtual instance.
- The ability to create, destroy, upgrade, and maintain Kubernetes clusters.
- Capable of deploying high-availability (HA) Kubernetes masters.
- Provisions the required cloud infrastructure on supported platforms. Currently, AWS is the only supported cloud provider, but support for other providers is available in either alpha or beta stages of development.
- The ability to generate Terraform configurations that can then be reviewed and deployed using terraform plan and terraform apply.
    - This feature is very useful because you can diff your configs in Git to see changes that are proposed or committed.
    - It is somewhat limited in that Kops still considers what it deploys to be the desired state, whereas Terraform configurations are considered a representation.
    - Therefore, if you change your Terraform configs and apply them directly, you may run into conflicts. In other words, it's probably best to make your changes through Kops rather than making them manually through Terraform unless you are an expert.
- Support for managed add ons, including:
    - AWS Load Balancer Controller, which helps support AWS ALB and NLBs.
    - Cluster autoscaler for adjusting the size and number of the worker nodes.
    - Cert-manager, a native Kubernetes certificate management controller used for managing certs from different providers such as Let's Encrypt, Vault, or self-signed certs, etc.
    - Metrics Server for collecting Kubernetes metrics via an API server for the purpose of cluster autoscaling. This feature is not meant for metrics forwarding or use with various monitoring or observability systems such as Prometheus.
    - Other features.
- Container Network Interface (CNI) is supported for different networking providers including AWS VPC, Flannel, Kubenet, Kube-router, Weave, etc.

Probably the biggest benefit of Kops is the granularity and flexibility one gains when using it with Kubernetes, learning all of the parts that one wouldn't normally learn with a hosted or managed service.

Probably the biggest benefit of Kops is the granularity and flexibility one gains when using it with Kubernetes, learning all of the parts that one wouldn't normally learn with a hosted or managed service.

However, it also requires a significantly higher overhead since there is a learning curve associated with getting up to speed on Kubernetes setup, configuration, and maintenance. Therefore, it might not be a feasible approach considering available personnel, in-house skill sets, and time to deliver.

Setup and Deployment

The first step in setting up and deploying a cluster with Kops is to install Kops and kubectl. You may also install other tools such as minikube and kubeadm. These are relatively simple operations that can be performed by almost anyone who is familiar with the command line on any OS. It can be done via brew on macOS, curl on Linux or macOS, or via binary on Windows.

Configuration and Access

The following is an example of configuration and access using AWS since it is the only supported cloud platform.

- You must have an AWS account set up and configured, and it should be configured using the recommended best practices.
- You should not use the unified billing or organizations master account or the root user in any AWS account to perform the actions required to set up your Kubernetes cluster. Instead, you should use a sub-account with a named IAM user and least privileges.

It takes about 30 minutes to get the necessary tools installed and run the required commands when you follow the steps listed on the Deploying to AWS page in the Kops documentation.

This is not a deep dive on Kops setup and configuration and there are several other steps involved including:

- Installation of AWS CLI tools, which is the standard practice for any type of CLI access to AWS.
- Creation of a dedicated Kops IAM user, group, and associated policies for CLI access to your designated AWS account.
- Configuration of Route53 to add required DNS records.
- Setup and configuration of an S3 bucket for storing cluster state.
- Creation of the cluster configuration using the Kops CLI.
- Building the cluster itself using the Kops CLI, which will create all the necessary AWS resources such as VPC, EC2, and EBS.

It takes about 30 minutes to get the necessary tools installed and run the required commands when you follow the steps listed on the Deploying to AWS page in the Kops documentation. After that, it takes about 10 minutes for Kops to instantiate all the components, including VPC, ASGs, EC2, and EBS volumes for the master and worker nodes, plus etcd.

You may experience a hiccup in the form of an "unauthorized" error when you try to get the cluster status using this command:

```
kops validate cluster
```

This may be due to changes in the authentication mechanism in Kubernetes v1.19. To correct this, try running the following command:

```
kops export kubecfg --admin
```

Thereafter, commands to check the status of the cluster should resemble:

```
○ → kops validate cluster
Using cluster from kubectl context: testcluster001.example.
com
Validating cluster testcluster001.example.com
INSTANCE GROUPS
NAME ROLE MACHINETYPE MIN MAX SUBNETS
master-us-west-2a Master t3.medium 1 1 us-west-2a
nodes-us-west-2a Node t3.medium 1 1 us-west-2a
NODE STATUS
NAME ROLE READY
ip-172-20-35-166.us-west-2.compute.internal master True
ip-172-20-52-15.us-west-2.compute.internal node True
Your cluster testcluster001.example.com is ready
```

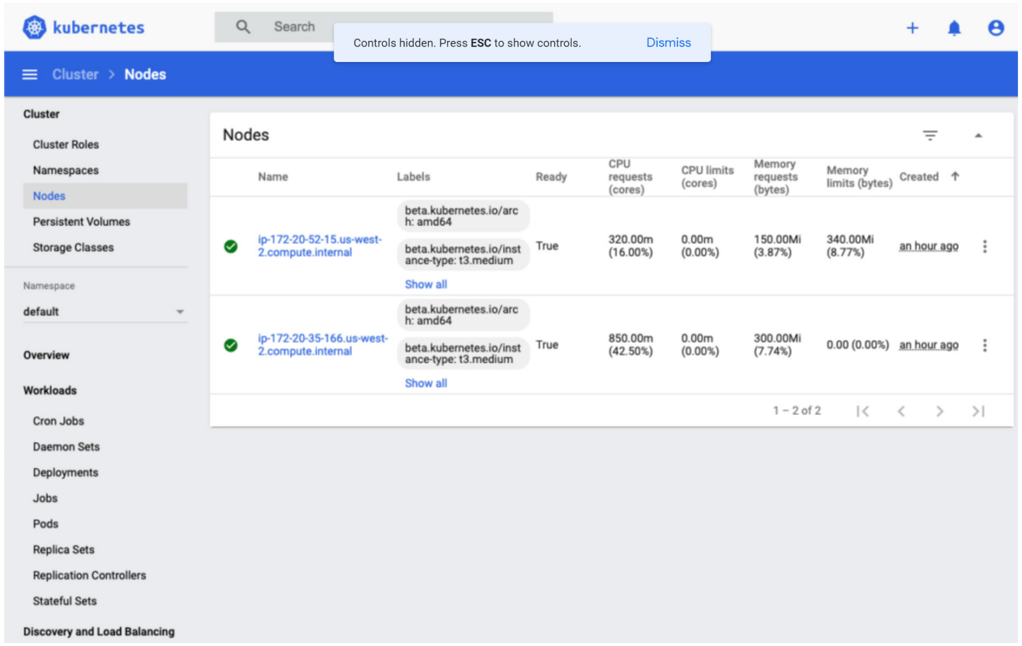
Kubernetes Dashboard requires additional configuration beyond just deploying it to the cluster in order to properly handle user authentication and make it available on localhost.

Cluster Management

Once you've completed the steps above, you can manage the cluster using kubectl or the Kops CLI itself depending on which actions you want to perform.

You may also install the Kubernetes Dashboard to perform some of the tasks normally done from the CLI such as checking cluster status or managing application deployments. It doesn't cover the full range of features that you get from the CLI tools, but it's a good start.

Dashboard requires additional configuration beyond just deploying it to the cluster in order to properly handle user authentication and make it available on localhost. This is covered in the documentation.



Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Created
ip-172-20-52-15.us-west-2.compute.internal	beta.kubernetes.io/arch: amd64; beta.kubernetes.io/instance-type: t3.medium; Show all	True	320.00m (16.00%)	0.00m (0.00%)	150.00Mi (3.87%)	340.00Mi (8.77%)	an hour ago
ip-172-20-35-166.us-west-2.compute.internal	beta.kubernetes.io/arch: amd64; beta.kubernetes.io/instance-type: t3.medium; Show all	True	850.00m (42.50%)	0.00m (0.00%)	300.00Mi (7.74%)	0.00 (0.00%)	an hour ago

1 – 2 of 2

If the Dashboard isn't suitable for your needs, there are other options for a Web UI with varying feature sets such as Octant or Lens.

When you are finished with your cluster, you can delete it using the Kops CLI:

```
kops delete cluster --name ${NAME} --yes
```

Kops delete output can be considerable and isn't included here. Any AWS resources (such as EC2 and EBS) that were created to support the cluster will be deleted when running the `kops delete cluster` command.

Portability

Any application that runs on a standard Kubernetes cluster will work when configuring a cluster with Kops since it uses upstream Kubernetes.

Kops does not support bare metal or on-prem VMs for deploying clusters. It currently supports the following cloud providers:

- AWS (officially supported)
- Digital Ocean, GCE, and OpenStack (in beta)
- Azure and Alicloud (in alpha)

While a number of Kubernetes networking providers are in different stages of development, only Calico, Cilium, and kubenet are currently stable. All others are still considered experimental.

Networking

Depending on how you configure your cluster using Kops, it will take care of any basic networking tasks required to spin up a new cluster. Load balancing is supported through Elastic Load Balancing.

While a number of Kubernetes networking providers are in different stages of development, only Calico, Cilium, and kubenet are currently stable. All others are still considered experimental.

Available features and limitations are determined by the network provider that you choose. For example, when using the default provider, kubenet, there is effectively a limit of 50 nodes due to the AWS limit of 50 routes in the routing table. To use kubenet, there are limited workarounds such as requesting an AWS limit increase.

Though kubenet is the default provider for Kops, due to its inherent limitations it is not recommended for production clusters that are expected to grow significantly.

EKS

EKS is AWS's implementation for hosted Kubernetes and is often the first stop for teams new to Kubernetes. It's a good option if you are looking to start out with Kubernetes and are already using AWS.

Since EKS exists within the AWS ecosystem, services like EC2, ELB, VPC, and Fargate are fully supported. AWS boasts a number of other features and integrations as well.

Features

AWS offers many features and options for EKS and is a great choice for getting up and running relatively quickly.

- Supports upstream Kubernetes and is certified Kubernetes conformant. Kubernetes applications can be ported to EKS without code modification.
- Managed, high-availability control plane runs across multiple AWS Availability Zones (AZs) with 99.95% SLA. This includes Kubernetes masters as well as etcd.
- Automatically replaces unhealthy control plane nodes.
- Managed, automatic security patching for the control plane.
- Supports AWS Fargate (serverless) as well as EC2 (for worker nodes).
- Supports EC2 spot instances for cost optimization.
- Includes console to view status and manage Kubernetes clusters.
- Ability to run Windows and Linux worker nodes side-by-side in the same cluster.
- Full Application Load Balancer (ALB), Network Load Balancer (NLB), and VPC support.
- Supports hybrid deployments (such as on-prem and cloud-based Kubernetes deployments) using a combination of AWS-hosted EKS alongside EKS Distro and EKS Anywhere (which hasn't been released yet) for on-prem, bare metal, and VMs.
- Supports Kubeflow with EKS for machine learning on AWS GPU instances.
- Kubernetes Jobs API for scheduled batch jobs.
- Support for Elastic Fabric Adaptor (EFA) for High-Performance Computing (HPC) workloads.
- Other features.

Drawbacks

Drawbacks to using EKS depend on your specific use case. Using EKS will limit you to using other AWS services to support it, if only for the sake of convenience. Each AWS service has its own advantages and disadvantages over similar offerings from other providers.

Cost can be another drawback. Each AWS service that you leverage usually has a cost component associated with it. These costs can add up quickly if you're not careful, so you will need to monitor them closely to avoid going over budget when using additional AWS services.

Cost can be another drawback. Each AWS service that you leverage usually has a cost component associated with it. These costs can add up quickly if you're not careful, so you will need to monitor them closely to avoid going over budget when using additional AWS services.

Amazon provides an open source version of Kubernetes called EKS Distro, which can be used to install EKS on bare metal, VMs, or EC2. The advantage of this option is that you can use the AWS Console or API for management purposes without needing to use third-party dashboards or tooling.

Setup and Deployment

There are a variety of options for setting up EKS including the AWS Console, cloudformation, Terraform, or eksctl, which uses cloudformation under the hood. There are pros and cons to using each.

To set up a test cluster with eksctl, follow the instructions in the AWS EKS documentation. The following macOS example does not include installing prerequisite AWS CLI and kubectl. It assumes you are an IAM user with administrative privileges in your AWS test account.

```
brew install eksctl
```

In this example, AWS Fargate is the option chose for worker nodes.

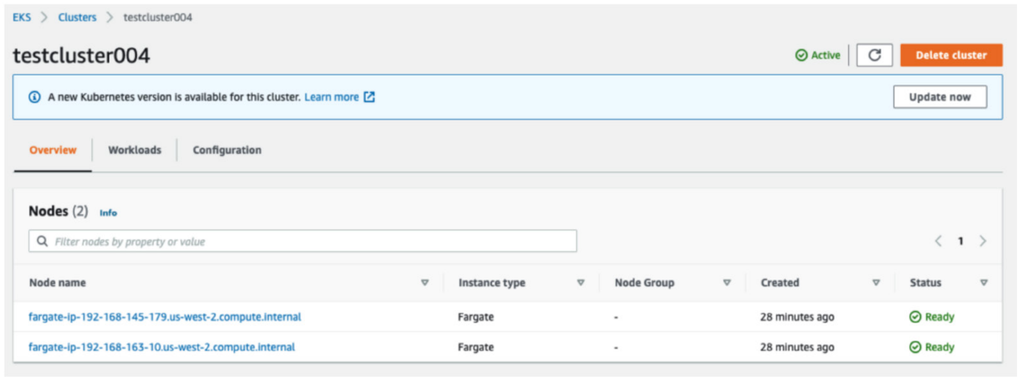
To instantiate the cluster:

```
eksctl create cluster --name testcluster004 --region us-
west-2 --fargate
```

The cluster should be up and running in about 15 minutes.

```
○ → kubectl get nodes
NAME              STATUS    ROLES     AGE      VERSION
fargate-ip-192-168-145-179.us-west-2.compute.internal   Ready
<none>    2m5s    v1.18.9-eks-866667
fargate-ip-192-168-163-10.us-west-2.compute.internal    Ready
<none>    2m13s   v1.18.9-eks-866667
```

It would appear like this in the AWS Console:


EKS > Clusters > testcluster004
Active
Delete cluster
Update now
Overview
Workloads
Configuration
Info
Filter nodes by property or value

testcluster004

A new Kubernetes version is available for this cluster. Learn more

Nodes (2)

Node name	Instance type	Node Group	Created	Status
fargate-ip-192-168-145-179.us-west-2.compute.internal	Fargate	-	28 minutes ago	✓ Ready
fargate-ip-192-168-163-10.us-west-2.compute.internal	Fargate	-	28 minutes ago	✓ Ready

Configuration and Access

EKS may ask you to update your version of Kubernetes, which typically takes an hour.

eksctl takes care of several steps such as designating the IAM user that created the cluster as the cluster administrator. If you plan on having a multi-user cluster, then you will need to configure additional IAM users.

There are additional steps to be performed in order to fully configure the cluster, including configuring the cluster autoscaler and preparing to deploy sample workloads.

Cluster Management

The EKS control plane is managed for you, so little is required for this once the cluster is instantiated. You will, however, have to manage worker nodes via EC2 or Fargate.

AWS has complete documentation for most of the other cluster management actions that you may perform such as installing Metrics Server, Prometheus, and the Kubernetes Dashboard.

To delete the cluster:

```
eksctl delete cluster --name my-cluster --region region
```

Portability

Amazon provides an open source version of Kubernetes called EKS Distro, which can be used to install EKS on bare metal, VMs, or EC2. The advantage of this option is that you can use the AWS Console or API for management purposes without needing to use third-party dashboards or tooling.

Using EKS and ancillary services limits you to AWS-related products and distributions whether on-prem or in the AWS cloud. This can be an advantage for teams that don't want to deal with the complexities of managing external vendors, services, or packages. On the other hand, it's a disadvantage if you're looking for portability or flexibility across multiple providers.

Networking

EKS supports Virtual Private Cloud (VPC) and related constructs as well as Application Load Balancer (ALB) and Network Load Balancer (NLB) via the AWS Load Balancer Controller. CoreDNS and Calico are also supported.

Using EKS and ancillary services limits you to AWS-related products and distributions whether on-prem or in the AWS cloud. This can be an advantage for teams that don't want to deal with the complexities of managing external vendors, services, or packages. On the other hand, it's a disadvantage if you're looking for portability or flexibility across multiple providers.

Platform9

Platform9 KaaS is certified Kubernetes conformant. It provides a cloud-hosted management plane and interface where all cluster management is performed.

Infrastructure can be deployed to the platform of choice including bare metal, VMs, or cloud providers like Azure, GCP, and AWS. Tasks such as cluster deployment, management, monitoring, and upgrading can all be done within Platform9.

Since Platform9 is a managed product, many of the problematic aspects of Kubernetes management can be offloaded to Platform9 — which can be a huge advantage. Features that would normally need to be built and configured by a Kubernetes administrator are provided as part of Platform9's offerings.

A more complete overview of Platform9 features and architecture can be found in the Platform9 documentation.

Features

- Fully managed SaaS platform for Kubernetes.
- Uses upstream Kubernetes.
- Supports Ubuntu and CentOS/RHEL.
- Zero downtime for cluster upgrades.
- Deployable on-prem to bare metal or VMs.
- Deployable to cloud providers such as Azure, GCP, and AWS.
- Single view and management of multiple Kubernetes clusters.
- Deploys highly available multi-master clusters.
- Etcd backups.
- CNI and CSI support.
- Monitoring and logging.
- CLI, Web UI, and REST API.
- Works with Terraform.
- Supports multiple region deployments.
- Multi-tenancy and multi-user support.
- SAML SSO.
- Provides up to 24x7x365 support via email, phone, or live chat and video, depending on your plan.
- 99.9% SLA for paid plans.

Since Platform9 is a managed product, many of the problematic aspects of Kubernetes management can be offloaded to Platform9 — which can be a huge advantage. Features that would normally need to be built and configured by a Kubernetes administrator are provided as part of Platform9's offerings.

It also makes cluster instantiation and maintenance tasks much easier. This frees up in-house resources for important engineering functions such as CI/CD, the Software Development Lifecycle (SDLC), and providing a stable platform for business applications.

Setup and Deployment

Platform9 can be deployed anywhere, including on fully managed bare metal or with VMs via KubeVirt. The following example recounts using AWS and Web UI to deploy a cluster using a Platform9 free account.

After signing up for a free account, we logged into into the dashboard to create a new cluster. We were presented with several options and chose Amazon Web Services with the one-click cluster setup option.

We created a Platform9 IAM user in the AWS Console and attached the policy provided by Platform9 directly to that user. There are a number of options to configure permissions for production.

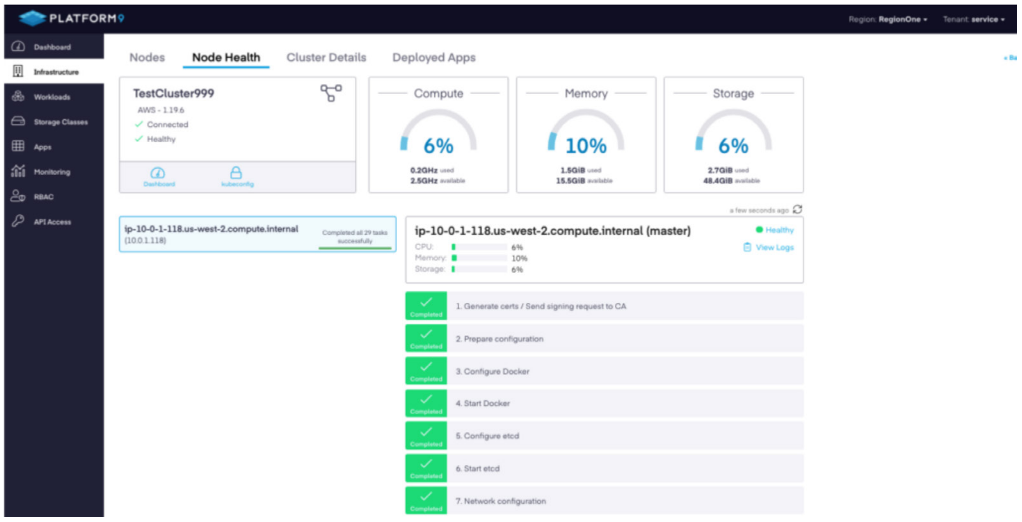
We copied the AWS Access Key and Secret Key for the new IAM user into the Platform9 Web UI and tested for access. From there, we continued with the cluster creation process.

To complete the cluster creation, we set a cluster name, cloud provider (AWS), region, and availability zone (us-west-2a). An SSH key and Route53 domain name were also required.

Platform9 currently supports Kubernetes versions up to 1.19.6 and we used that version for this cluster.

Finally, we clicked "create cluster." It took a few minutes for the console to update, and then we were presented with a list of items that were being deployed for our cluster. An abbreviated representation appears below.

Platform9 can be deployed anywhere, including on fully managed bare metal or with VMs via KubeVirt. The following example recounts using AWS and Web UI to deploy a cluster using a Platform9 free account.



TestCluster999
AWS - 1.19.6
✓ Connected
✓ Healthy

Compute 6% 0.2GHz used 2.5GHz available
Memory 10% 1.5GiB used 15.5GiB available
Storage 6% 2.7GiB used 48.4GiB available

a few seconds ago

ip-10-0-1-118.us-west-2.compute.internal (10.0.1.118) Completed all 29 tasks successfully

ip-10-0-1-118.us-west-2.compute.internal (master) ● Healthy
CPU: 6%
Memory: 10%
Storage: 6%

✓ Completed 1. Generate certs / Send signing request to CA
✓ Completed 2. Prepare configuration
✓ Completed 3. Configure Docker
✓ Completed 4. Start Docker
✓ Completed 5. Configure etcd
✓ Completed 6. Start etcd
✓ Completed 7. Network configuration

Configuration and Access

The configuration process for using Platform9 on AWS is straightforward and requires:

- An AWS account that is not your master billing or AWS Organizations account.
- An IAM user for Platform9. This can be a pre-existing or newly created IAM user. You should note the Access Key and Secret Key, since you will need them when you add the AWS cloud provider to Platform9 via the console.
- Platform9 provides a downloadable IAM policy which you will need to attach directly to the IAM user. Better yet, we recommend adding that user to a group and then attaching the policy to the group.
- An EC2 keypair created before you add the cloud provider in Platform9.

We hadn't yet created one when we added the AWS cloud provider to Platform9, so it didn't show up. Adding the EC2 keypair, then deleting and adding the cloud provider again via the console fixed this issue.

- A region with available EC2 instances.
- A Route53 domain name that's already configured. Since we have a test domain for these purposes, that was already there for us.

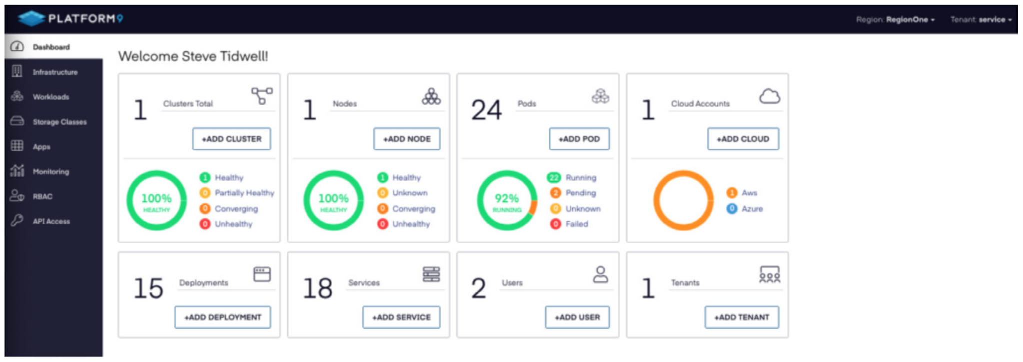
With the above in place, add the cloud provider and select the requisite options described above. Adding a cluster is then quite easy.

Cluster Management

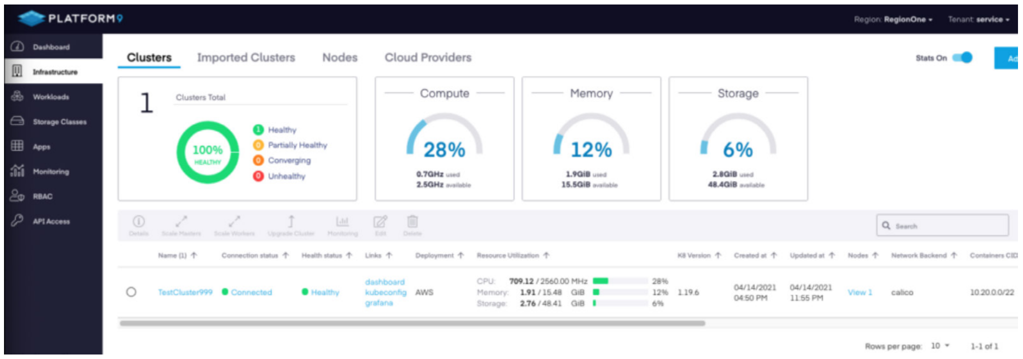
Cluster management via the Platform9 console is intuitive. All actions such as adding a cloud provider, cluster, node, or an application are done through the UI.

Deleting the cluster via the UI is just as simple, and the Dashboard provides comprehensive visibility:

Since Platform9 uses upstream Kubernetes and is certified Kubernetes conformant, any application deployed on Kubernetes will work when using Platform9. Since it has multicloud support, you can centrally manage all of your Kubernetes clusters across all providers including on-prem, bare metal, and VMs.



Drilling down to the new cluster itself shows more details:



	Name (1)	Connection status	Health status	Links	Deployment	Resource Utilization	K8 Version	Created at	Updated at	Nodes	Network Backend	Containers CID
○	TestCluster999	Connected	Healthy	dashboard kubeconfig grafana	AWS	CPU: 709.12 / 2560.00 MHz 28% Memory: 1.91 / 15.48 GiB 12% Storage: 2.76 / 48.41 GiB 6%	1.19.6	04/14/2021 04:50 PM	04/14/2021 11:55 PM	View 1	calico	10.20.0.0/22

Rows per page: 10 1-1 of 1

Portability

Since Platform9 uses upstream Kubernetes and is certified Kubernetes conformant, any application deployed on Kubernetes will work when using Platform9. Since it has multicloud support, you can centrally manage all of your Kubernetes clusters across all providers including on-prem, bare metal, and VMs.

Lock-in isn't an issue since Kubernetes is deployed into your own cloud accounts or onto your own servers or VMs. Applications are portable between clusters on different providers.

Networking

Platform9 comes with managed CNI and Calico is the preferred plugin.

VPC, Route53, and Elastic Load Balancers are all supported on AWS. If you're running on-prem, Platform9 supports MetalLB for load balancing with bare metal.

Conclusion

Evaluating various Kubernetes deployment and management solutions can be complicated. It's very important to keep your use case and requirements in mind when looking for one. It will take time to evaluate, set up, configure, and test each solution. Last, but certainly not least, you want to consider the learning curve involved with each solution compared to the knowledge, skills, and resources that you have in-house.

Platform9 KaaS — a simple way to get started with fully-managed Kubernetes clusters.

Platform9 KaaS — a simple way to get started with fully-managed Kubernetes clusters.

Platform9 is free for up to 20 nodes. Start a free trial today!
info@platform9.com