Kubernetes on AWS: Tutorial and K8s Install

If you're looking for information on how to get started with Kubernetes on AWS Cloud, you've come to the right place. In this document we will share with you what we've learned over the years and what we recommend for setting up and running Kubernetes on AWS. We'll provide you with useful resources, both practical and theoretical, so that you can avoid some of the pitfalls along the way.

Who we are

As the creators of GitOps, we help teams adopt and manage cloud-native infrastructure and applications quickly, securely, reliably, and at scale. Through our GitOps solutions, we enable infrastructure, and application development teams to build and operate their own Kubernetes

Further Reading

Kubernetes Library

Learn More ❯

CI/CD for Kubernetes

Learn More ❯

Best Practices for Hybrid Cloud Kubernetes with EKS

Download Now ❯

application platform whether in the cloud, at the edge, or on-premise. Our products and solutions are community-built and enterprise-approved. Weaveworks was one of the first members of the Cloud Native Computing Foundation and is one of its top 10 contributors.

AWS has partnered with Weaveworks because we have more experience operating Kubernetes at scale than any other independent company. Not only are we an ISV Partner and an Advanced Technology Partner, but AWS is one of our strategic investors as well; furthermore solidifying their belief in the continued innovation of our products.

AWS not only trusts Weaveworks to work with some of their largest strategic accounts, such as Apple, Fidelity and HSBC, but they also work directly with us on their core products.

Weaveworks created EKSctl, a tool used by AWS customers and employees.

aws partner network

Advanced
Technology
Partner

Container Competency

In addition to this, we are major contributors to the Kubernetes Open

Source project; originators of the Kubernetes on AWS SIG; and we're also key members of the SIG Cluster Lifecycle.

Accelerate Kubernetes Adoption with GitOps Design Patterns

With our experience, we can help you navigate the challenges of running Kubernetes on AWS. Together we build a well architected Kubernetes and operate workloads across various underlying infrastructures such as on-premise, vSphere, AKS, GKE, EKS or others. Weave GitOps Kubernetes Design Patterns feature Cluster API or Terraform techniques, empowers organizations to build fully declarative, continuously reconciled and multi-tenant Kubernetes platforms following GitOps best practices.

Contact us today for a no cost evaluation or see our offering on the AWS Marketplace.

Why Cloud Native Tools?

Cloud Native is open source cloud computing for applications—a trusted tool kit for modern architectures.

FROM THE BLOG

EKS Blueprints and GitOps best practices

Read More ❯

GitOps With GitHub Actions and EKS

Read More ❯

eksctl and Flagger

Read More ❯

Weaveworks is a founding member of the Cloud Native Computing Foundation (CNCF) and we believe the future is cloud native.

We use Docker containers and manage them in Kubernetes clusters for all of the same reasons that have led you to containers and Kubernetes. Containers are lightweight, portable and they allow you to make fast incremental changes, which ultimately provides more value more quickly to your customers—even more so if you're using a microservices-based architecture.

If you're managing those containers with Kubernetes—a project started and used by Google—you know that you can easily scale your application without having to worry about rebuilding the cluster.

As developers we like that mostly hands-off approach. We'd rather spend our time coding without having to worry too much about the infrastructure on which it runs. The more we worry about infrastructure, the fewer features we produce and this is generally not a good thing in today's competitive landscape.

But Why Run Kubernetes on AWS?

AWS is a premier solution for running cloud native apps, but setting up and running Kubernetes on it can be complex. Despite this, there are many reasons to run Kubernetes on AWS.

One of the most appealing reasons is to take advantage of the vast number of services that are available. Other reasons to run Kubernetes on AWS, over say, ECS include:

- Complete control over your servers — An advantage of using Kubernetes on AWS is that it puts you in control over your instances which is not always the case with other cloud providers.

- Access to Open Source Software without Vendor Lock-in — Kubernetes is completely open source and so are many of the tools surrounding the project. This provides you with a wide-open, well-supported community with many options.

- Portability — Kubernetes runs anywhere: bare metal, public cloud, private cloud, and can even run on multiple public clouds all at once if you wish.

- Cloudbursting and Private workload protection — With Kubernetes, you can run part of your cluster in the public cloud, but then have sensitive workloads that spill over and run in a private cloud on-premises, for example.

When you're installing Kubernetes on AWS, these are the services that you will need to be familiar with. In each section, we describe what you need to know when you're configuring a cluster.

But before we get into the details of each Amazon service and how they apply to Kubernetes, it is useful to have some familiarity with the Kubernetes architecture and its parts. See the interactive tutorial, "Kubernetes Basics" for a good overview.

Amazon Elastic Kubernetes Service (EKS)

For those of you who don't want to manage every aspect of Kubernetes yourselves, you can use the Amazon Elastic Kubernetes Service (EKS). This hosted EC2 service takes away most of the heavy lifting of manual configuration so that you can easily run Kubernetes on AWS by providing:

- Managed Kubernetes control plane.
- Highly Available cluster.
- Automated version upgrades.
- Integration with some AWS services like CloudTrail,CloudWatch, ELB,IAM, VPC, PrivateLink.

For an easy and quick installation of Kubernetes on AWS try the open source tool eksctl and with only one command have a fully functional Kubernetes cluster running in AWS' EKS in minutes.

Links:

- eksctl – open source tool for setting up a cluster on EKS

AWS Services and Kubernetes

Amazon VPC

Amazon Virtual Private Cloud (VPC) service lets you provision private, isolated sections of the AWS Cloud and then launch AWS services and other resources onto a virtual network. With a VPC you can define your own IP address range and have complete control over your virtual networking environment, including subnets, and route table definition as well as network gateways.

VPC networking vs. Kubernetes networking

One of the concepts that may be confusing is the networking. There are a few different networks that you need to be aware of when you're running Kubernetes in AWS.

A VPC has its own networking capabilities and it connects cluster nodes or EC2 instances to each other onto its own subnet. A Kubernetes cluster also has its own network—a pod network—which is separate from a VPC instance network.

Pods are collections of containers with shared storage/network with a specification for how to run the containers. Pods are generally co-located, and co-scheduled and they run in a shared context. This means that containers within pods share an application model and can also share components through local volumes between related services within an application. Each pod has its own IP that are managed and scheduled by the Kubernetes master node.

But pods between EC2 instances need a way to communicate with each other. The VPC itself provides support for setting routes through the kubenet plugin (deprecated as of 1.8). This is a very basic Linux networking plugin that provides near-native performance throughput for your cluster but it lacks other advanced features such as extensive networking across availability zones, the ability to enforce a security policy and also when using a VPC, you cannot effectively network the cluster since it uses multiple route tables. This is why many people resort to using CNI plugins -- an open standard for container communications.

CNI Plugins

Container Network Plugins (CNI) for Kubernetes provide a lot more features than the basic `kubenet linux` networking plugin does. While you do lose some performance with a CNI overlay network, you gain other things like being able to set security policy rules between your services as well as the ability to connect nodes and pods between high availability (HA) zones if you have a cluster that is larger than 50 nodes.

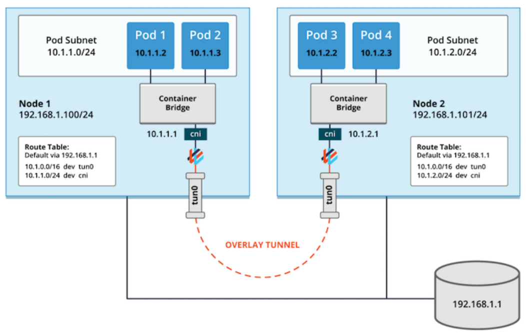
During installation you can specify which CNI plugin you want to use for the pod network. There are several network plugins available: Weave Net (and specifically for EKS) Calico, and Flannel and others.

For a good discussion on CNI, why you need it and a comparison of the different CNI providers, see “Choosing a CNI Network Provider for Kubernetes”.

Out of these plugins Weave Net is the best option for a number of reasons. See “Pod Networking in Kubernetes” for more information.

Weave Net has been implemented by operations and development teams on almost every public and private cloud including Alibaba Cloud, Amazon Web Services, Google Cloud Platform, Microsoft Azure, Oracle Cloud, Red Hat OpenShift and VMware Tanzu (previously Pivotal Container Service - PKS).

Weaveworks is part of Amazon Web Services’ network of partners that offer alternative CNI plugins for Elastic Kubernetes Service (EKS). Learn more about how to configure Weave Net with EKS.



Amazon EC2

Amazon Elastic Compute Cloud provides scalable secure instances within a VPC. You can provision a virtual instance with any operating system by choosing one of the many Amazon Machine Images (AMIs) available or create your own AMI for distribution and for your own use.

EC2 Nodes & Kubernetes

When creating instances for your cluster you'll need to think about the size of the nodes. Even though Kubernetes automatically scales and adjusts to a growing app, the resources set for any EC2 nodes you initially create are static and they cannot be changed afterwards.

Scaling Nodes

Scaling nodes is not supported through Kubernetes' command-line interface, `kubectl` in AWS. If you need to use Kubernetes autoscaler, then you'll need to do it manually through the AWS with the Autoscaling Group feature or you can also manually create a set number of EC2 nodes to achieve the same result.

Amazon Route53 for Kubernetes Cluster Setup

Kubernetes clusters need DNS so that the worker nodes can talk to the master as well as discover the etcd and then the rest of its components.

When running Kubernetes in AWS, you can make use of Amazon Route 53 or you can run an external DNS.

If you will be running multiple clusters, each cluster should have its own subdomain as well.

Load Balancers

There are basically two design patterns in AWS where you may need load balancers:

- During the installation of Kubernetes on AWS

- When you’re exposing app services to the outside world and you have deployed more than one master running, you may need to provision an external load balancer so that you have an externally-accessible IP address for your application that is accessible to the outside world.

For more information about finding and exposing an external IP for Kubernetes see the section below on How to Define Ingress and for more in depth information refer to the topic, Publishing

Services in the Kubernetes documentation.

How to Define Ingress in AWS

Ingress is not a service in AWS and its rules must be defined separately for any of your app's services that need to be exposed to the outside world.

Setting up Ingress in AWS involves the following:

- Transport Layer Security (TLS) certification
- host-names
- path endpoints (optional)
- services and service ports

When running Kubernetes on AWS, there are a few different ways to handle ingress:

- Deploy an ingress controller provided through the Kubernetes API
- You can integrate the Zalando Kube-ingress controller which is an AWS-native solution
- Use a built-in controller with an external load balancer like NGINX
- Define a load balancer with TLS for each service you want to expose in the Kubernetes manifest and then use `kubectl expose service_name`.

Network Policy & Security

Related to ingress is the ability to specify a security network policy for every service available in a pod and whether it's accessible to the outside world or to another service.

If you are using Weave Net as your CNI pod networking layer, then you will have a Network Policy available to you, and when configured, Weave Net will enforce that policy. Network policies are very easily specified in the kubernetes deployment manifests (YAML files).

For information on how to do that, see "What is a Network Policy Controller?" and "Configuring a Network Policy".

Amazon EBS

Amazon Elastic Block Store (Amazon EBS) provides persistent block storage volumes for use with EC2 cloud instances. Each Amazon EBS volume is automatically replicated within its Availability Zone to protect you from component failure, offering high availability and durability. Amazon EBS volumes provide consistent and low-latency performance needed to run your workloads.

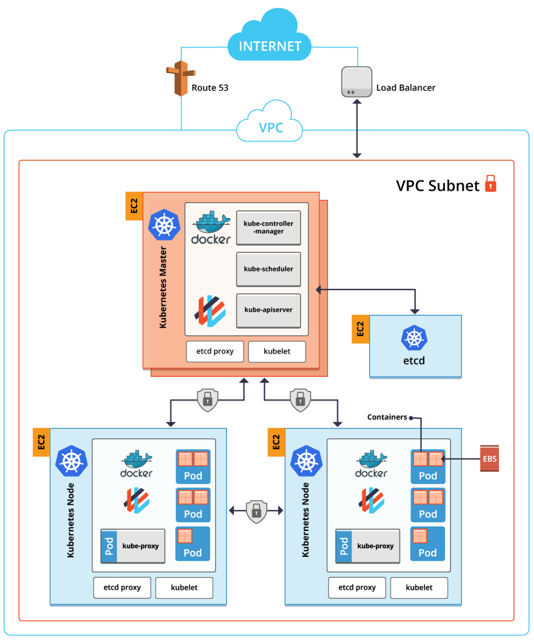
We recommend using Amazon EBS with Kubernetes if you require a backup for any services that are already backed up with Kubernetes persistent volumes.

Data Stores and Kubernetes

Sometimes pods need persistent data across volumes. For example if some of your containers are MySQL databases (or any databases for that matter), and they crash, having a backup for your persistent volumes ensures that when the MySQL container comes back up, it can resume where it left off.

See the discussion on volumesfor information on how Kubernetes manages data stores and Persistent Volumes for available parameters.

Basic Kubernetes with AWS Setup



Identity & Access Management (IAM)

Kubernetes does not provide specific AWS IAM roles and permissions. If you are storing and retrieving information from an S3 Bucket or from DynamoDB (calls the AWS API directly), then you will need to think about how to provide IAM permissions for your nodes, pods, and containers. Normally you will want different IAM roles for the masters and the nodes.

You could assign a global IAM role to a Kubernetes node, where all of the IAM roles required by all containers and pods running in Kubernetes are automatically inherited. But from a security standpoint, this is not an optimal. Instead, you will need a more granular approach, one that can assign IAM roles at the Pod and the container level and not just at the node level.

If you use kops to set up your cluster two IAM roles are set up for your cluster one for the masters and one for the nodes.

There are a few different approaches to manage the AWS security requirements:

- Group authentication models for applications on Kubernetes and then give groups of nodes certain IAM permissions
- Implement a proxy such as kube2iam
- Use a solution like vaultto handle app-level secrets

Summary of Kubernetes on AWS

At a high level, these are the issues you need to consider when running Kubernetes on AWS:

AWS Service	Why you need it & what to consider
Amazon VPC	• Use a CNI network for HA clusters with > 50 nodes • Incorporate capacity planning for node resources
EC2	• Nodes can't be scaled through `kubectl`; needs the autoscaling feature either in the GUI or not
Amazon Route53	• Kubernetes clusters require DNS to discover all of its components • Multiple clusters need subdomains
Ingress Rules & Elastic Load Balancer	• Ingress rules definition & planning • Use Amazon's LB • NGINX or use the ingress controller

	provided by the with Kubernetes API
Amazon EBS	• Allocate elastic block storage for stateful applications to ensure continuity during downtime
Identity & Access Management (IAM)	• Kubernetes controller needs IAM roles for master and nodes • May need finer grain control if you are accessing the AWS API directly

With our experience, we can help you navigate the challenges of running Kubernetes on AWS. Together we build a well architected Kubernetes and operate workloads across various underlying infrastructures such as on-premise, vSphere, AKS, GKE, EKS or others. Weave GitOps Kubernetes Design Patterns feature Cluster API or Terraform techniques, empowers organizations to build fully declarative, continuously reconciled and multi-tenant Kubernetes platforms following GitOps best practices.

CONTACT US FOR NO COST EVALUATION