SKILLS | DevOps and Cloud Computing

Scaling Kubernetes and Infrastructure as Code (IaC) in DevOps

Objective

- Understand how Kubernetes scales worker nodes dynamically.
- Learn about high availability strategies in Kubernetes, including multiple master nodes and failover handling.
- Explore Infrastructure as Code (IaC) and how it automates infrastructure deployment and versioning.
- Identify how IaC fits into DevOps workflows and integrates with CI/CD pipelines.

Explaining how Kubernetes automatically scales worker nodes based on demand

Let’s see

Kubernetes automatically scales worker nodes based on demand using a component called the Cluster Autoscaler. Here’s how it works in short:

- The Cluster Autoscaler continuously monitors the cluster for resource shortages—specifically, it looks for pending pods that cannot be scheduled because existing nodes lack sufficient CPU or memory resources.
- When the autoscaler detects that there are unschedulable pods due to insufficient capacity, it automatically provisions (adds) new worker nodes to the cluster so the pods can be scheduled and run.

Let’s see

- Conversely, if it finds that some nodes are underutilized (i.e., not needed for current workloads), it can automatically remove (scale down) those nodes to optimize costs and resource usage.
- This process ensures that the cluster always has enough capacity to handle workload spikes, while also minimizing unnecessary infrastructure during low-demand periods.

PW SKILLS

Discussing Cluster Autoscaler and how it integrates with cloud-based Kubernetes clusters (AWS, GCP, Azure).

Let’s discuss

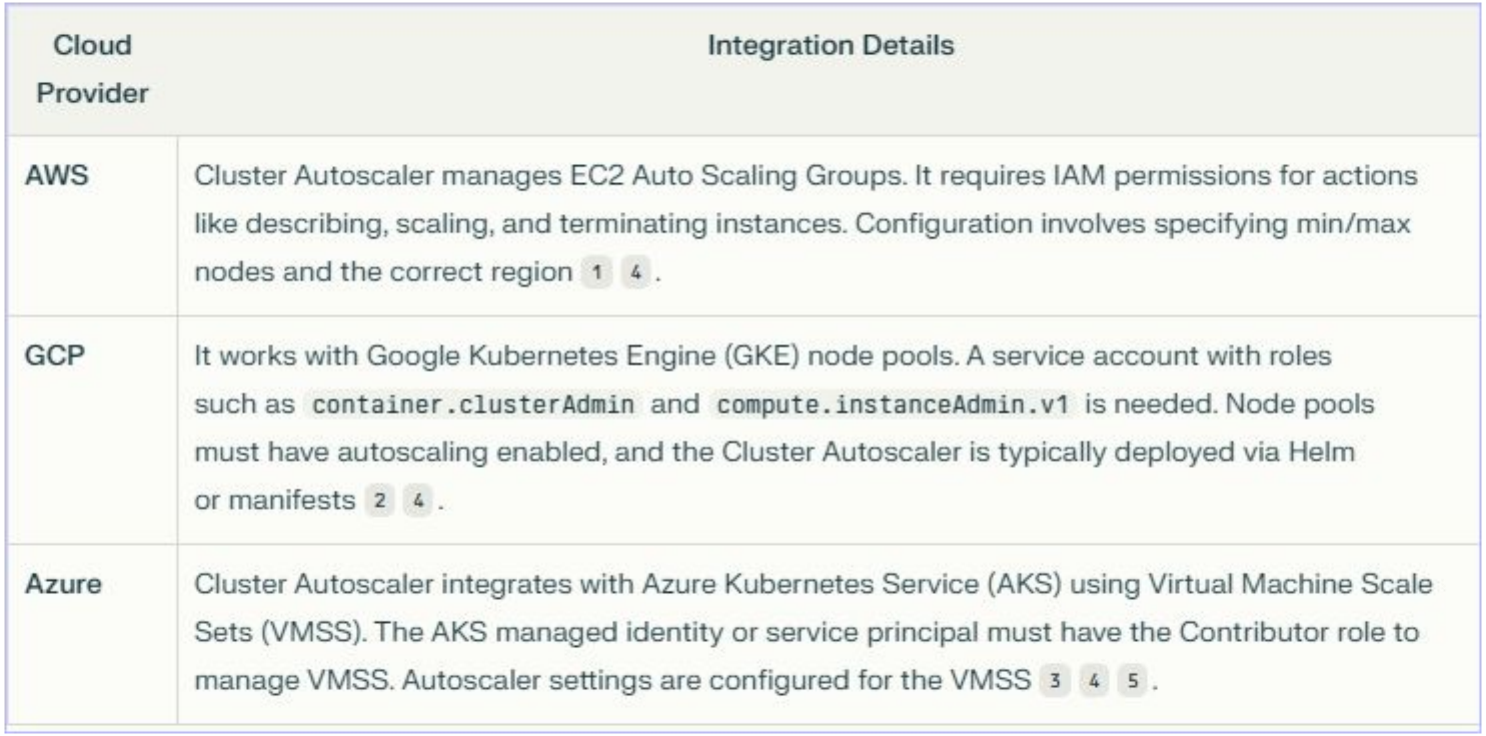
The Cluster Autoscaler is a Kubernetes component that automatically adjusts the number of worker nodes in a cluster based on resource demand.

- When pods can't be scheduled due to insufficient resources, it adds nodes; when nodes are underutilized, it removes them to save costs.

Integration with Cloud Providers

Cluster Autoscaler integrates seamlessly with major cloud platforms—AWS, GCP, and Azure—by interacting with their respective infrastructure APIs to manage node groups or scale sets:

Let’s discuss



Cloud Provider	Integration Details
AWS	Cluster Autoscaler manages EC2 Auto Scaling Groups. It requires IAM permissions for actions like describing, scaling, and terminating instances. Configuration involves specifying min/max nodes and the correct region [1] [4].
GCP	It works with Google Kubernetes Engine (GKE) node pools. A service account with roles such as `container.clusterAdmin` and `compute.instanceAdmin.v1` is needed. Node pools must have autoscaling enabled, and the Cluster Autoscaler is typically deployed via Helm or manifests [2] [4].
Azure	Cluster Autoscaler integrates with Azure Kubernetes Service (AKS) using Virtual Machine Scale Sets (VMSS). The AKS managed identity or service principal must have the Contributor role to manage VMSS. Autoscaler settings are configured for the VMSS [3] [4] [5].

PW SKILLS

Explaining the benefits of dynamic scaling in cloud-native environments.

Let’s see

Dynamic scaling in cloud-native environments offers several key benefits:

- Optimal Performance
- Cost Efficiency
- Agility and Flexibility
- Resilience and Reliability
- Reduced Operational Overhead

Pop Quiz

Q. What happens if one control plane node fails and a load balancer is present?

A

The etcd cluster is deleted

B

The load balancer reroutes traffic to healthy nodes

Pop Quiz

Q. What happens if one control plane node fails and a load balancer is present?

A

The etcd cluster is deleted

B

The load balancer reroutes traffic to healthy nodes

Explaining why high availability (HA) is critical for production environments.

Let’s see

High availability (HA) is critical for production environments because it ensures systems remain operational and accessible even in the event of hardware failures, software issues, or other disruptions.

- This minimizes unplanned downtime, which can otherwise lead to significant revenue loss, production delays, customer dissatisfaction, and damage to brand reputation.
- HA is achieved by eliminating single points of failure through redundancy and failover mechanisms, allowing essential business processes and services to continue without interruption.

PW SKILLS

Discussing multiple master node setups and their role in failover handling.

Let’s discuss

Multiple master node setups in Kubernetes ensure high availability (HA) by distributing the control plane across several nodes, eliminating single points of failure.

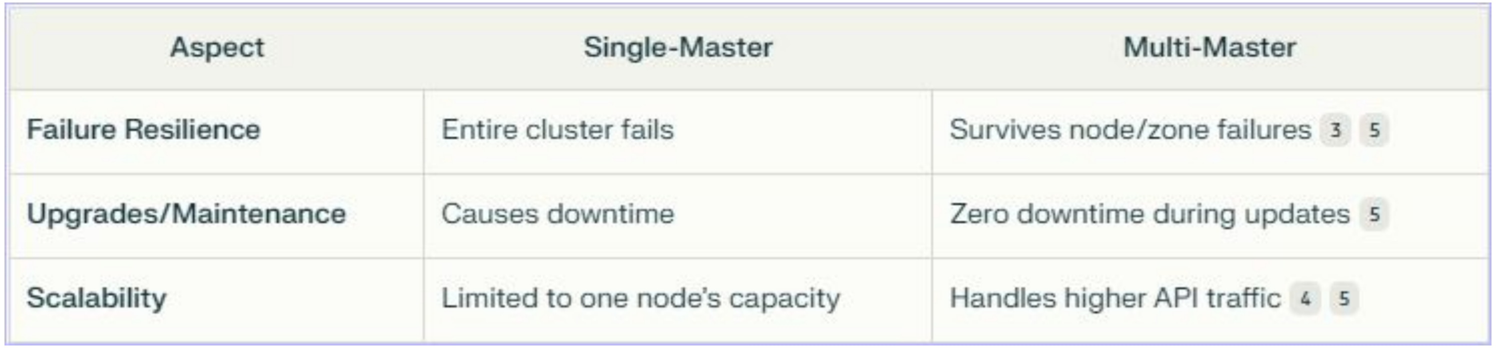
This configuration enables automatic failover, maintaining cluster operations even if a master node becomes unavailable.

Key Components and Failover Handling:

- Control Plane Redundancy
- etcd Clustering
- Load Balancer Integration

Let’s discuss

Benefits of Multi-Master Setups



Aspect	Single-Master	Multi-Master
Failure Resilience	Entire cluster fails	Survives node/zone failures [3] [5]
Upgrades/Maintenance	Causes downtime	Zero downtime during updates [5]
Scalability	Limited to one node’s capacity	Handles higher API traffic [4] [5]

Pop Quiz

Q. In a multi-master Kubernetes setup, what happens if one master node fails?

A

The remaining master nodes continue to manage the cluster without interruption

B

The API server becomes unavailable

Pop Quiz

Q. In a multi-master Kubernetes setup, what happens if one master node fails?

A

The remaining master nodes continue to manage the cluster without interruption

B

The API server becomes unavailable

PW SKILLS

Explaining how etcd cluster redundancy ensures data consistency across master nodes.

Let’s see

etcd cluster redundancy ensures data consistency across master nodes by replicating data among multiple etcd nodes using the Raft consensus algorithm. In this setup, one node acts as the leader, handling all write requests, while the others are followers.

- When a write occurs, the leader proposes the change to followers and waits for a majority (quorum) to confirm before committing the update.
- This guarantees that all nodes agree on the same data state, even during node failures or network partitions, so querying any node returns consistent results.
- This approach provides strong consistency, high availability, and fault tolerance for Kubernetes cluster state management.

PW SKILLS

Introducing load balancing for HA control planes in cloud and on-premises deployments.

Load balancing

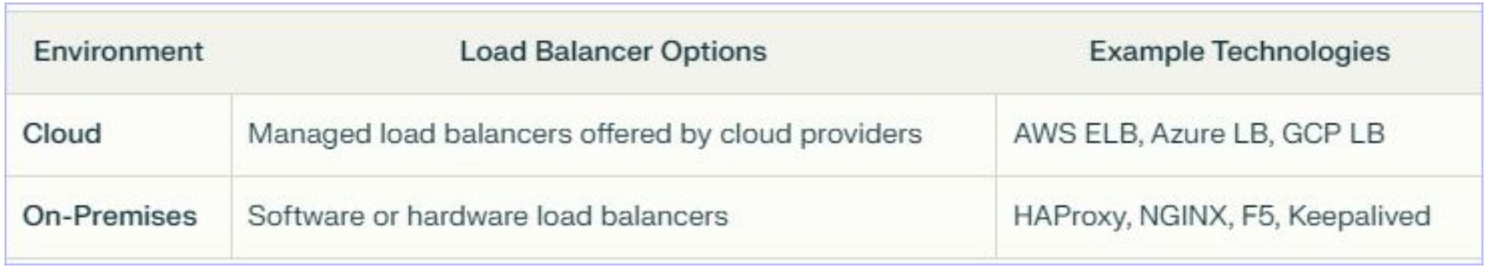
Load balancing is a fundamental requirement for high availability (HA) in Kubernetes control planes, ensuring seamless access and resilience whether deployed in the cloud or on-premises.

How Load Balancing Supports HA Control Planes:

- Distributes Traffic
- Automated Failover
- Consistent Access

Load balancing

Deployment Options:



Environment	Load Balancer Options	Example Technologies
Cloud	Managed load balancers offered by cloud providers	AWS ELB, Azure LB, GCP LB
On-Premises	Software or hardware load balancers	HAProxy, NGINX, F5, Keepalived

PW SKILLS

Take A 5-Minute Break!

- Stretch and relax
- Hydrate
- Clear your mind
- Be back in 5 minutes

PW SKILLS

Explaining how IaC replaces manual infrastructure provisioning with automated scripts.

Let’s see

Infrastructure as Code (IaC) replaces manual infrastructure provisioning by allowing IT teams to define and manage infrastructure using automated scripts or code, rather than performing repetitive tasks by hand.

- With IaC, you write code that specifies the desired state of servers, networks, and other resources; tools then automatically create, configure, and maintain these resources as defined.
- This automation speeds up deployment, ensures consistency across environments, reduces human error, and makes it easy to version, test, and reproduce infrastructure setups.

Discussing the benefits of consistency, versioning, and automation in IaC.

Benefits

Consistency, versioning, and automation are core benefits of Infrastructure as Code (IaC):

Consistency: IaC enables repeatable and standardized infrastructure deployments by codifying configurations.

- This eliminates configuration drift and human error, ensuring every environment is set up identically, which improves reliability and reduces troubleshooting time.

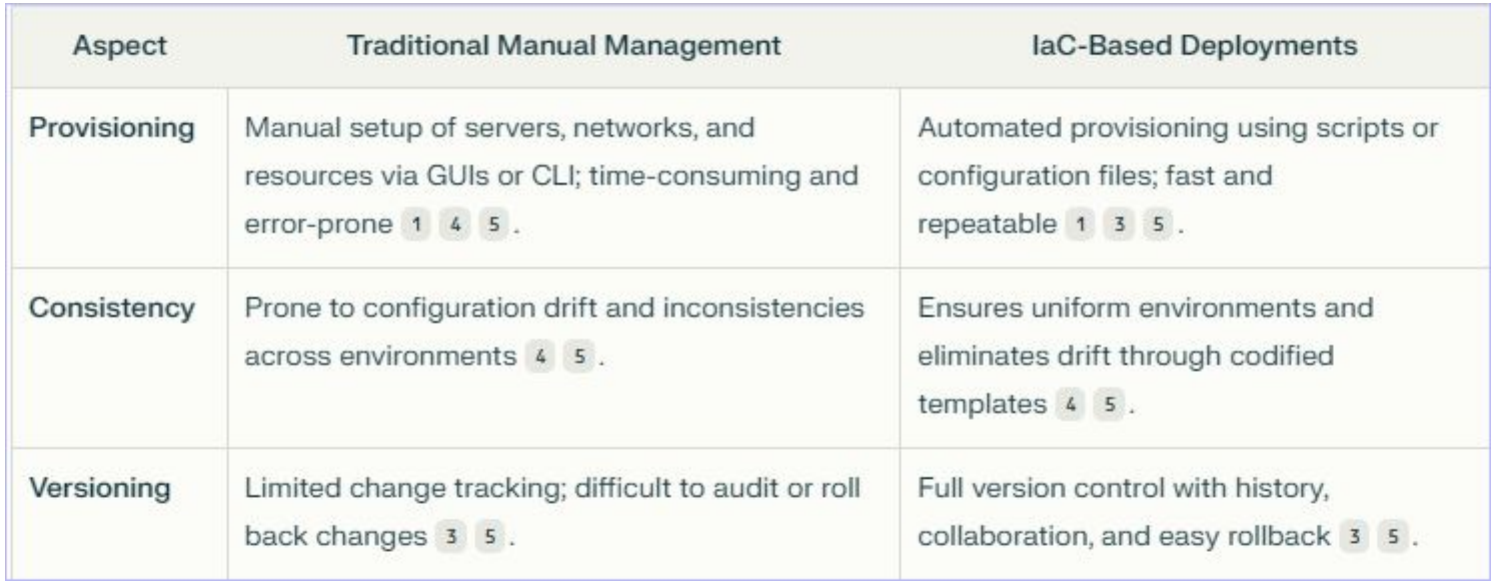
Versioning: Infrastructure definitions are stored as code in version control systems, allowing teams to track changes, audit updates, and roll back to previous states if needed. This enhances accountability and makes it easy to understand who changed what and when.

Benefits

Automation: IaC automates the provisioning and management of infrastructure, speeding up deployments and reducing manual effort. Automation not only increases operational efficiency but also minimizes the risk of errors and enables rapid scaling and recovery.

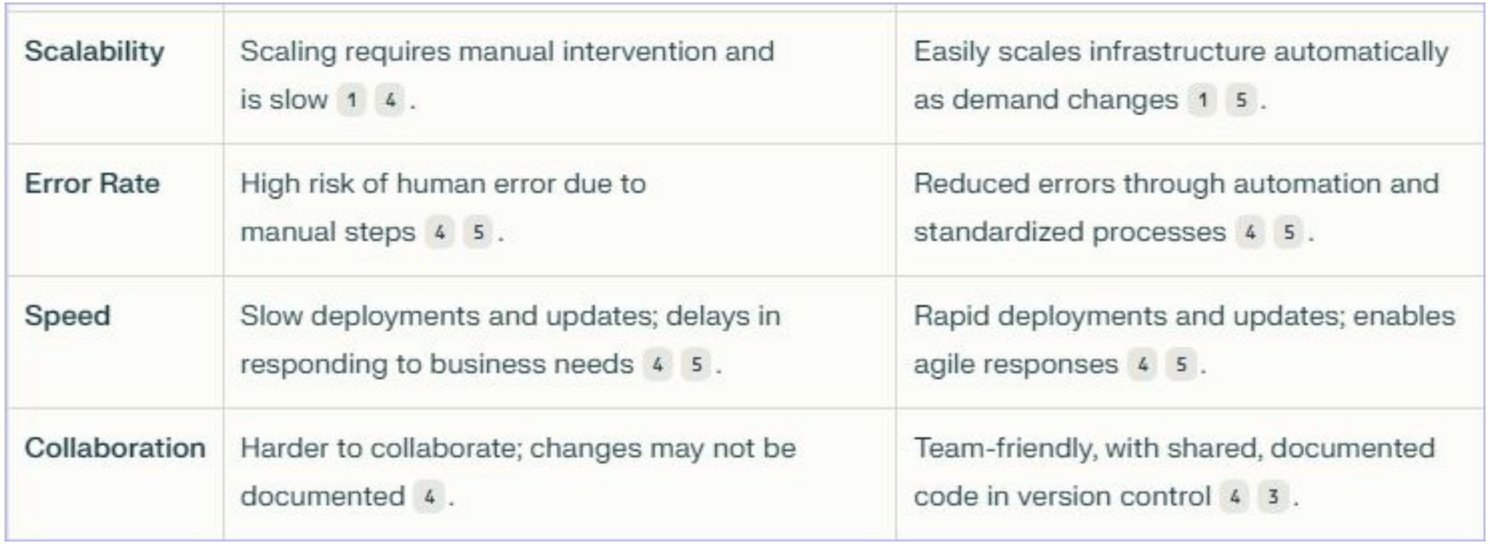
Comparing traditional manual infrastructure management vs. IaC-based deployments.

Let’s compare



Aspect	Traditional Manual Management	IaC-Based Deployments
Provisioning	Manual setup of servers, networks, and resources via GUIs or CLI; time-consuming and error-prone [1] [4] [5].	Automated provisioning using scripts or configuration files; fast and repeatable [1] [3] [5].
Consistency	Prone to configuration drift and inconsistencies across environments [4] [5].	Ensures uniform environments and eliminates drift through codified templates [4] [5].
Versioning	Limited change tracking; difficult to audit or roll back changes [3] [5].	Full version control with history, collaboration, and easy rollback [3] [5].

Let’s compare



Scalability	Scaling requires manual intervention and is slow [1] [4].	Easily scales infrastructure automatically as demand changes [1] [5].
Error Rate	High risk of human error due to manual steps [4] [5].	Reduced errors through automation and standardized processes [4] [5].
Speed	Slow deployments and updates; delays in responding to business needs [4] [5].	Rapid deployments and updates; enables agile responses [4] [5].
Collaboration	Harder to collaborate; changes may not be documented [4].	Team-friendly, with shared, documented code in version control [4] [3].

Explaining how IaC integrates with DevOps principles such as automation, testing, and continuous deployment.

Let’s see

Automation: IaC allows infrastructure to be defined and managed as code, enabling automated provisioning, configuration, and scaling of resources. This eliminates manual steps, speeds up deployments, and ensures consistency across environments.

Testing: IaC scripts can be version-controlled and subjected to automated testing, including security scans and compliance checks, just like application code. This ensures infrastructure changes are validated before reaching production, reducing errors and risks.

Continuous Deployment: IaC is integrated into CI/CD pipelines, allowing infrastructure changes to be automatically deployed alongside application updates. This supports rapid, reliable, and repeatable releases, aligning infrastructure management with modern DevOps workflows.

PW SKILLS

Discussing how IaC works with CI/CD tools (GitHub Actions, Jenkins, GitLab CI) for infrastructure deployment

Let’s discuss

Infrastructure as Code (IaC) works with CI/CD tools like GitHub Actions, Jenkins, and GitLab CI by automating the entire process of infrastructure deployment through pipelines. Here’s how the integration typically works:

- Version Control
- Automated Pipelines
- Testing and Compliance
- Consistent Deployments

PW SKILLS

Explaining the concept of GitOps and how version control systems manage infrastructure changes.

Let’s see

GitOps is an operational framework that uses Git as the single source of truth for managing infrastructure and application deployments.

- In GitOps, all infrastructure definitions and configurations are stored as code in a Git repository. Changes to infrastructure are made through pull requests and merge workflows, enabling version control, collaboration, and automated deployment processes.

Version control systems like Git manage infrastructure changes by:

- Tracking every change
- Enabling collaboration
- Supporting rollbacks
- Automating deployments

Explaining how IaC provisions and manages cloud resources (VMs, Kubernetes clusters, networking).

Let’s see

- Infrastructure as Code (IaC) provisions and manages cloud resources—such as virtual machines (VMs), Kubernetes clusters, and networking—by defining the desired state of these resources in code or configuration files.
- IaC tools interact directly with cloud provider APIs to automatically create, update, or delete resources as specified, eliminating manual setup and configuration.

For example:

- VMs
- Kubernetes Clusters
- Networking

Discussing state management and drift detection in IaC tools.

Let’s discuss

State management in Infrastructure as Code (IaC) tools involves tracking the current configuration of infrastructure resources in a "state file," which records what has been provisioned and how it aligns with the desired configuration defined in code.

Drift detection is the process by which IaC tools identify discrepancies between the desired state (as defined in code), the cached state (recorded in the state file), and the actual state (resources running in the cloud or data center).

PW SKILLS

Introducing version control for infrastructure, ensuring auditability and rollback options.

Let’s see

Version control for infrastructure involves storing configuration files, scripts, and infrastructure code in a version control system (VCS) like Git.

With version control, teams can:

- Auditability: Review the complete history of infrastructure changes, including who made each change and why, making it easier to investigate issues or comply with regulatory requirements.

Let’s see

- Rollback Options: Instantly revert to a previous, stable state if a new change introduces errors or instability, minimizing downtime and risk.
- Collaboration: Multiple team members can work on infrastructure code simultaneously, using branches and pull requests to review and merge changes safely.
- Consistency: Ensure all environments (dev, staging, production) are aligned by promoting tested and versioned infrastructure definitions through automated pipelines.

PW SKILLS

Time for case study!

Important

- Complete the post-class assessment
- Complete assignments (if any)
- Practice the concepts and techniques taught in this sessior
- Review your lecture notes
- Note down questions and queries regarding this session ar consult the teaching assistants

Thanks
!