Scaling Applications with Kubernetes using a Kubernetes Emulator (MiniKube)

UNIVERSITY *of* WASHINGTON

1

Introduction to Kubernetes

Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

- It groups containers that make up an application into logical units for easy management and discovery.

What does scaling applications with kubernetes mean?

Adjusting the number of instances (replicas) of an application or the resources allocated to those instances to handle changes in demand or workload. This ensures the application remains available, responsive, and efficient, whether demand increases (e.g., during traffic spikes) or decreases (e.g., during off-peak hours).

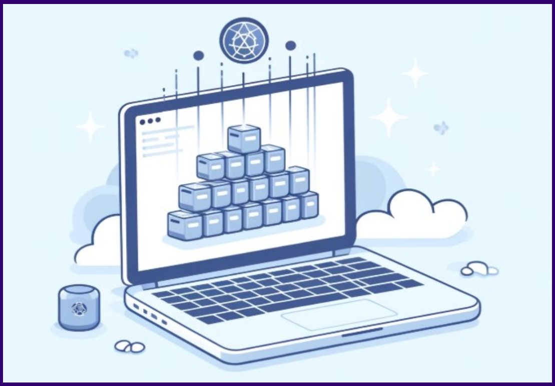
UNIVERSITY *of* WASHINGTON

2

Minikube: Simulating Application Scaling with Kubernetes

Minikube is a lightweight Kubernetes emulator that runs a single-node Kubernetes cluster locally on your laptop using a virtual machine. It enables developers to test and experiment with Kubernetes without needing a full cloud environment.

Using Minikube to simulate **scaling applications with Kubernetes** involves deploying an application on a local Kubernetes cluster and then testing scaling features such as adding replicas or adjusting resource limits. While Minikube runs on a single node (since it emulates Kubernetes locally), it can still demonstrate the concepts of scaling.



UNIVERSITY *of* WASHINGTON

3

THE MOTIVATION:

The design and development of **Kubernetes** was inspired by Google's Borg cluster manager and based on ***Promise Theory***. Many of its top contributors had previously worked on Borg they codenamed Kubernetes ***"Project 7"*** after the *Star Trek* ex-Borg character Seven of Nine and gave ***its logo a seven-spoked ship's wheel*** (designed by Tim Hockin). Unlike Borg, which was written in C++ Kubernetes is written in the ***Go*** language.

The idea for **Minikube** came from the desire to provide a **lightweight solution** that could allow developers to run a single-node Kubernetes cluster **locally** on their machines in an isolated, cost-effective way.

UNIVERSITY *of* WASHINGTON

5

Building Minikube: From Concept to Reality

To implement Minikube, the development team leveraged existing **virtualization technologies**:
•**VirtualBox**: For creating and managing the VM that would run Kubernetes locally.
•**Docker**: For containerization, allowing Kubernetes components to run inside containers.
•**Kubernetes' own features**: Minikube was designed to run Kubernetes just like it would on a production cluster, but in a scaled-down, single-node environment.
The goal was to use **virtual machines** (VMs) to run Kubernetes as a **self-contained environment**, making it easy for developers to spin up a local Kubernetes cluster with a few simple commands.

Building the Tool:
•**Automated Setup**: Minikube automates the setup process of creating a VM, installing Kubernetes, and configuring it to run locally. This was done using a set of scripts that handled the installation and configuration of Kubernetes on the VM.
•**Integration with Virtualization Platforms**: Minikube was built to work with different virtualization platforms, such as **VirtualBox**, **VMware**, and **Docker**, to allow users to choose the platform that best suited their environment.

UNIVERSITY *of* WASHINGTON

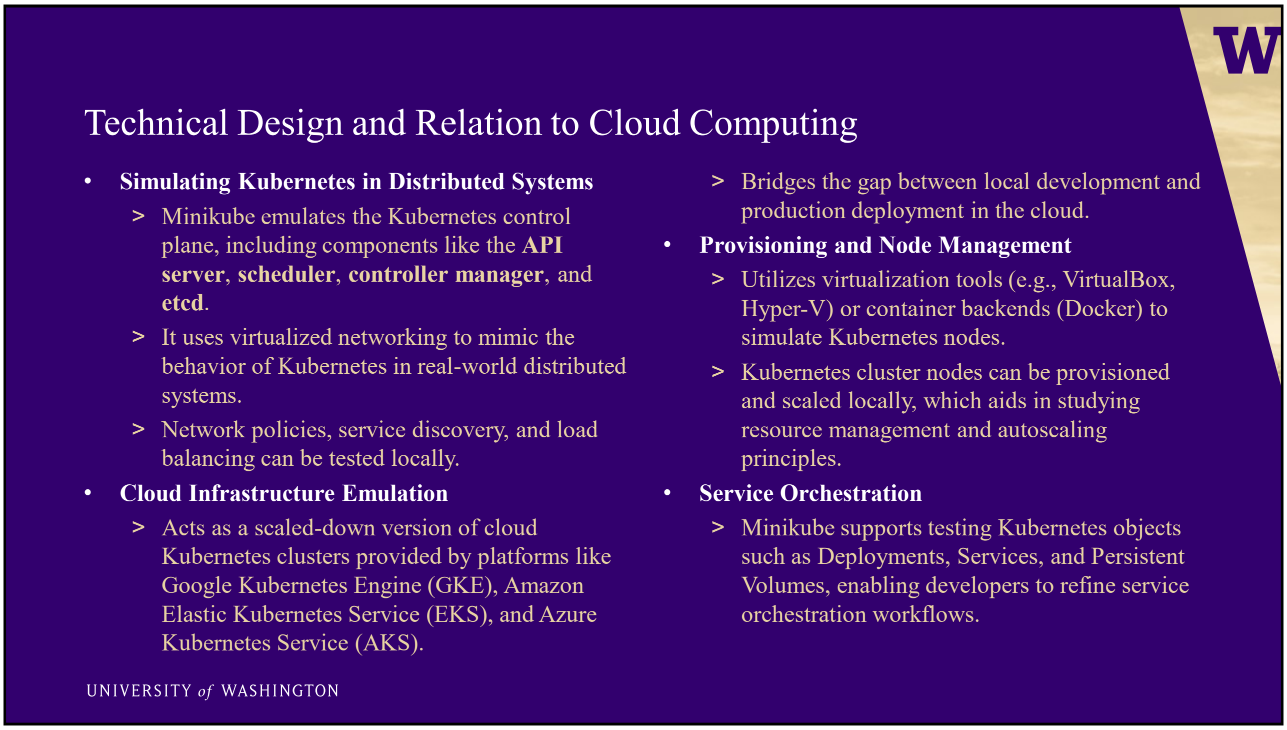
6

Exploring Scaling Features in Minikube

1. **Local Kubernetes Cluster**
Runs a Kubernetes cluster on a single node in a virtual machine (VM) or container on your local machine. Supports multiple operating systems (Linux, macOS, Windows).
2. **Multi-Node Clusters**
Allows creation of clusters with multiple nodes for simulating distributed systems.
3. **Container Runtime Support**
Supports Docker, CRI-O, and containerd as container runtimes, allowing users to test different setups.
4. **Hot Reloading**
Enables seamless code and configuration changes without restarting the cluster.
5. **Resource Efficiency**
Minimal resource requirements make it suitable for machines with limited computing power.
6. **Compatibility with CI/CD Pipelines**
Can be integrated into Continuous Integration and Continuous Deployment (CI/CD) pipelines to test Kubernetes configurations.

UNIVERSITY *of* WASHINGTON

7

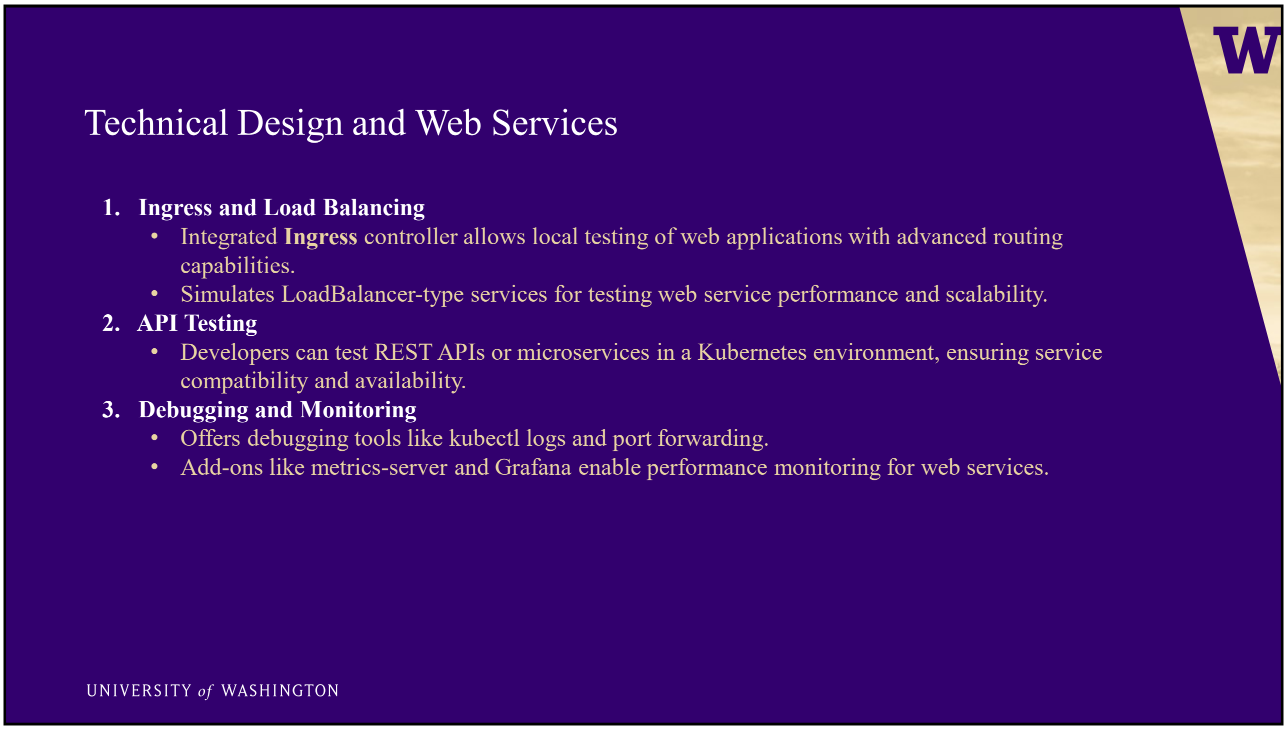


Technical Design and Relation to Cloud Computing

- **Simulating Kubernetes in Distributed Systems**
  - > Minikube emulates the Kubernetes control plane, including components like the **API server**, **scheduler**, **controller manager**, and **etcd**.
  - > It uses virtualized networking to mimic the behavior of Kubernetes in real-world distributed systems.
  - > Network policies, service discovery, and load balancing can be tested locally.
- **Cloud Infrastructure Emulation**
  - > Acts as a scaled-down version of cloud Kubernetes clusters provided by platforms like Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Service (EKS), and Azure Kubernetes Service (AKS).
  - > Bridges the gap between local development and production deployment in the cloud.
- **Provisioning and Node Management**
  - > Utilizes virtualization tools (e.g., VirtualBox, Hyper-V) or container backends (Docker) to simulate Kubernetes nodes.
  - > Kubernetes cluster nodes can be provisioned and scaled locally, which aids in studying resource management and autoscaling principles.
- **Service Orchestration**
  - > Minikube supports testing Kubernetes objects such as Deployments, Services, and Persistent Volumes, enabling developers to refine service orchestration workflows.

UNIVERSITY *of* WASHINGTON

8



Technical Design and Web Services

1. **Ingress and Load Balancing**
   - Integrated **Ingress** controller allows local testing of web applications with advanced routing capabilities.
   - Simulates LoadBalancer-type services for testing web service performance and scalability.
2. **API Testing**
   - Developers can test REST APIs or microservices in a Kubernetes environment, ensuring service compatibility and availability.
3. **Debugging and Monitoring**
   - Offers debugging tools like kubectl logs and port forwarding.
   - Add-ons like metrics-server and Grafana enable performance monitoring for web services.

UNIVERSITY *of* WASHINGTON

9

Use Cases in Real-World Scenarios

1. **E-Commerce**
Use Case: Handle high traffic during events like Black Friday.
Example Deployment: Test microservices, load balancing, and blue-green deployments locally to ensure smooth scaling under heavy load.
2. **Financial Services**
Use Case: Support secure, real-time trading systems.
Example Deployment: Simulate fault tolerance, validate resource policies, and test autoscaling for transaction-heavy services.
3. **Healthcare**
Use Case: Manage HIPAA-compliant patient data.
Example Deployment: Test persistent storage, encryption, and multi-tenant setups for healthcare platforms.
4. **Media Streaming**
Use Case: Scale video delivery systems dynamically.
Example Deployment: Simulate caching, autoscaling, and service mesh features to optimize user experiences during peak hours.

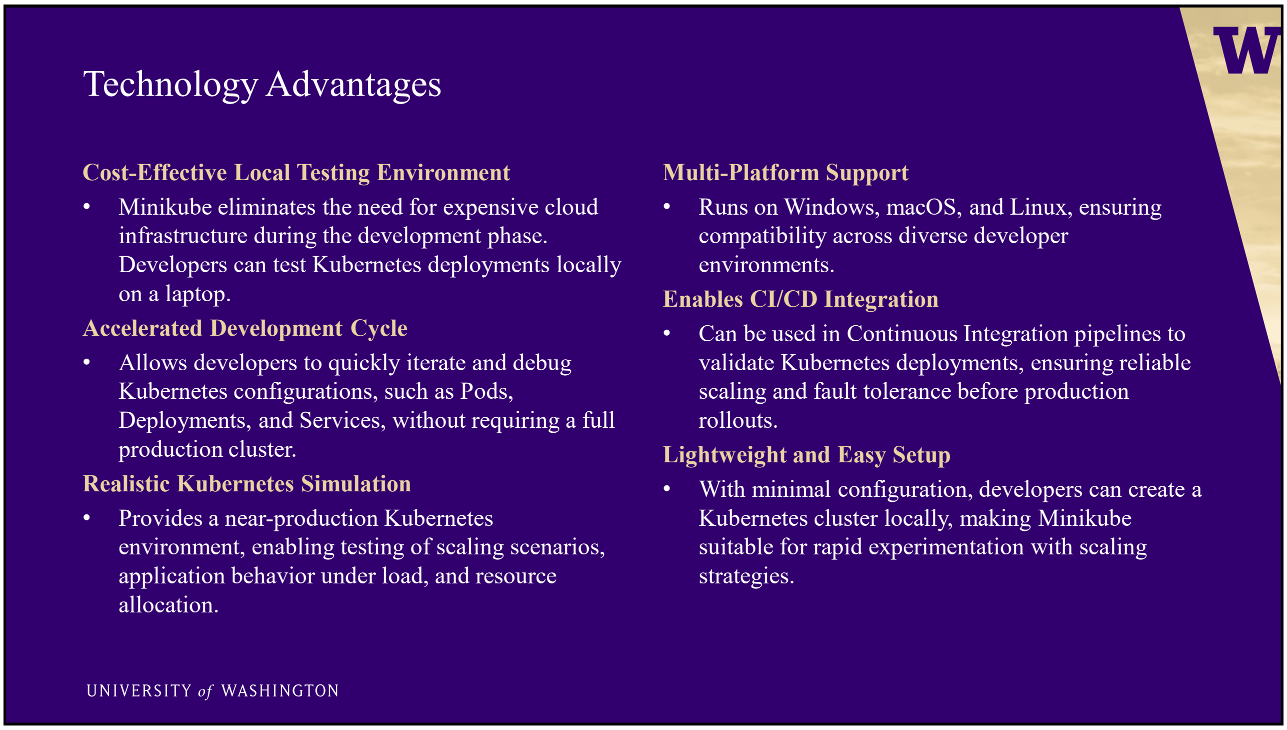
UNIVERSITY *of* WASHINGTON

10

5. **Gaming**
Use Case: Scale game servers for multiplayer games.
Example Deployment: Test server scaling, messaging systems, and traffic routing for real-time gameplay.
6. **SaaS Platforms**
Use Case: Scale collaboration tools dynamically.
Example Deployment: Test multi-region deployments, database scaling, and rolling updates for SaaS tools like Slack or Jira.
7. **Manufacturing and IoT**
Use Case: Process real-time IoT data on factory floors.
Example Deployment: Simulate edge computing, IoT pipelines, and workload scaling for smart manufacturing systems.
8. **AI and Machine Learning**
Use Case: Scale model training and inference workloads.
Example Deployment: Validate AI pipelines, GPU/TPU support, and autoscaling for real-time predictions.

UNIVERSITY *of* WASHINGTON

11



Technology Advantages

Cost-Effective Local Testing Environment

- Minikube eliminates the need for expensive cloud infrastructure during the development phase. Developers can test Kubernetes deployments locally on a laptop.

Accelerated Development Cycle

- Allows developers to quickly iterate and debug Kubernetes configurations, such as Pods, Deployments, and Services, without requiring a full production cluster.

Realistic Kubernetes Simulation

- Provides a near-production Kubernetes environment, enabling testing of scaling scenarios, application behavior under load, and resource allocation.

Multi-Platform Support

- Runs on Windows, macOS, and Linux, ensuring compatibility across diverse developer environments.

Enables CI/CD Integration

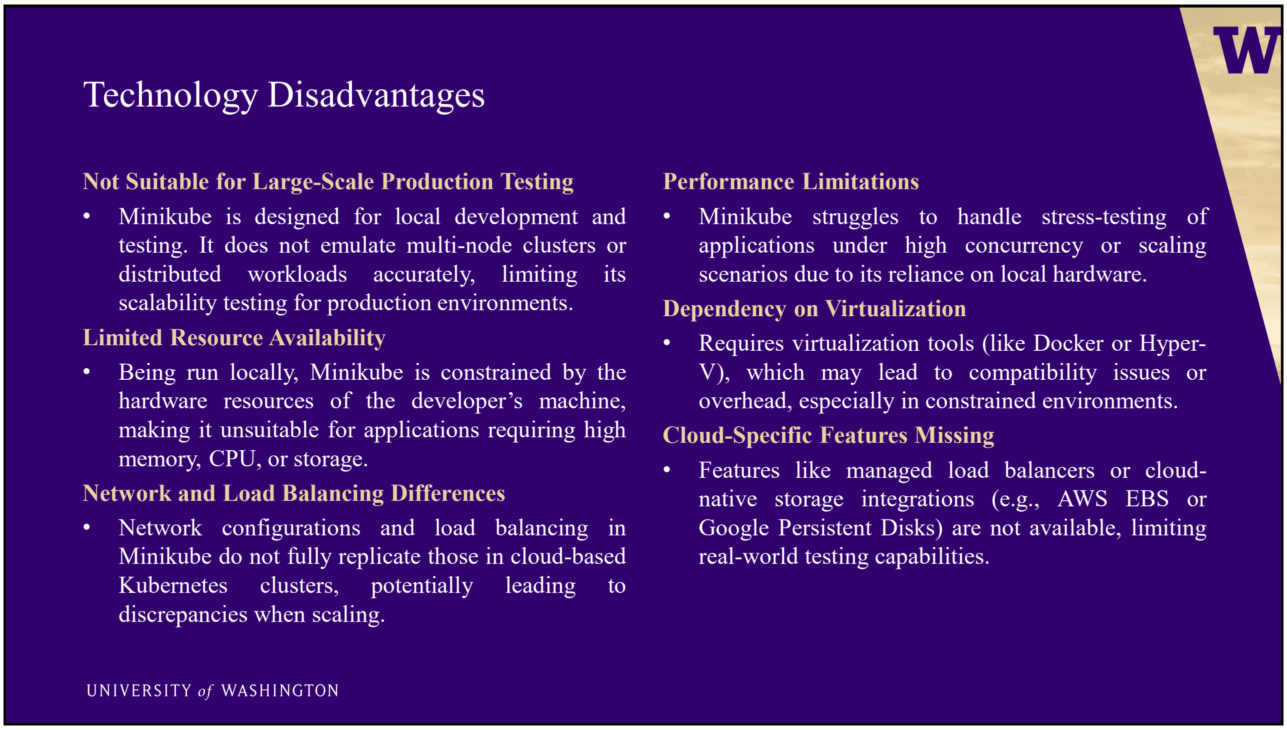
- Can be used in Continuous Integration pipelines to validate Kubernetes deployments, ensuring reliable scaling and fault tolerance before production rollouts.

Lightweight and Easy Setup

- With minimal configuration, developers can create a Kubernetes cluster locally, making Minikube suitable for rapid experimentation with scaling strategies.

UNIVERSITY *of* WASHINGTON

12



Technology Disadvantages

Not Suitable for Large-Scale Production Testing

- Minikube is designed for local development and testing. It does not emulate multi-node clusters or distributed workloads accurately, limiting its scalability testing for production environments.

Limited Resource Availability

- Being run locally, Minikube is constrained by the hardware resources of the developer's machine, making it unsuitable for applications requiring high memory, CPU, or storage.

Network and Load Balancing Differences

- Network configurations and load balancing in Minikube do not fully replicate those in cloud-based Kubernetes clusters, potentially leading to discrepancies when scaling.

Performance Limitations

- Minikube struggles to handle stress-testing of applications under high concurrency or scaling scenarios due to its reliance on local hardware.

Dependency on Virtualization

- Requires virtualization tools (like Docker or Hyper-V), which may lead to compatibility issues or overhead, especially in constrained environments.

Cloud-Specific Features Missing

- Features like managed load balancers or cloud-native storage integrations (e.g., AWS EBS or Google Persistent Disks) are not available, limiting real-world testing capabilities.

UNIVERSITY *of* WASHINGTON

13

Usability

Ease of Use
- Minikube is beginner-friendly with straightforward setup steps and detailed documentation.
- Ideal for developers and teams new to Kubernetes who want a hands-on learning experience.

Learning Curve
- Initial concepts like Pods, Services, and Ingress can be challenging for newcomers, but Minikube provides a sandboxed environment to experiment safely.

Programming APIs
- Seamlessly integrates with Kubernetes tools (kubectl, Helm), enabling developers to test applications with real-world APIs.

Interactive Tooling
- Minikube includes features like the dashboard, logs, and live resource monitoring to enhance usability during local development.

UNIVERSITY *of* WASHINGTON

14

Cost Discussion

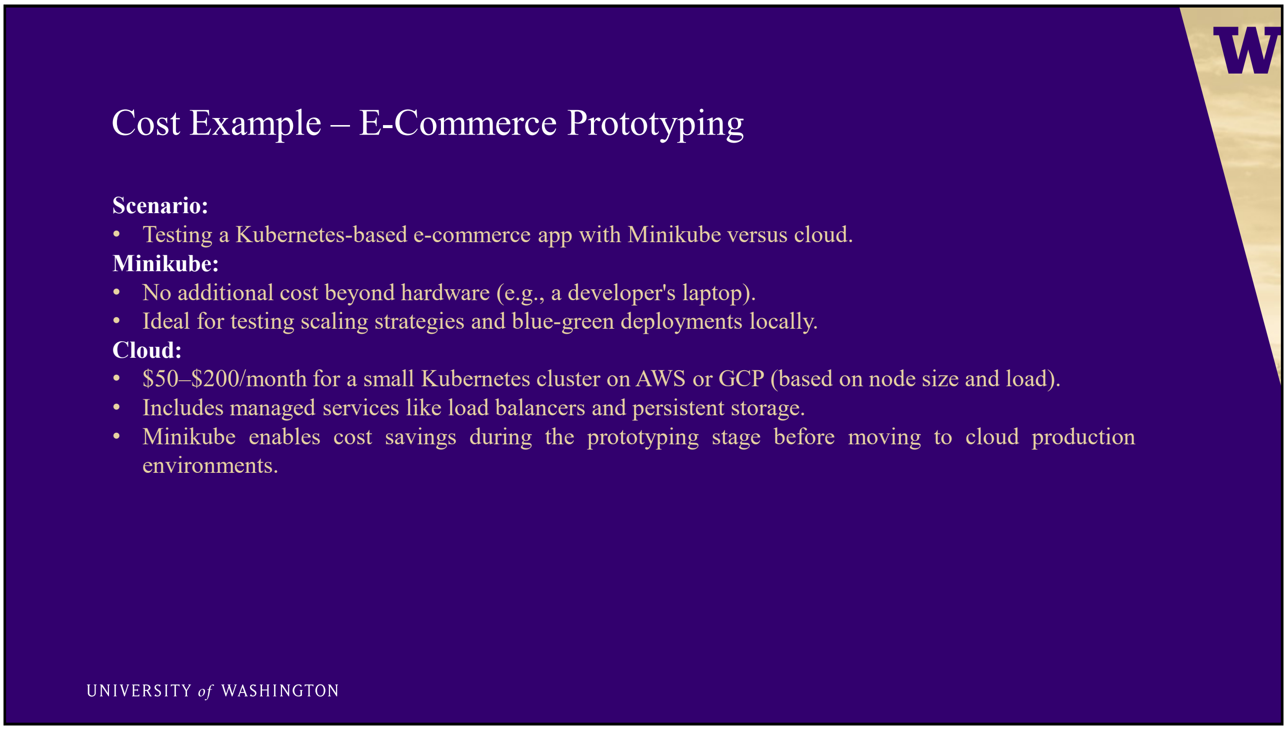
Minikube Costs
- Free to use as an open-source project, requiring only local hardware resources.
- Minimal infrastructure cost compared to cloud-hosted Kubernetes clusters.

Comparison with Cloud Kubernetes
- Running Kubernetes in the cloud involves ongoing costs for compute, storage, and network usage, often exceeding hundreds of dollars monthly for moderate workloads.
- Minikube offers a low-cost alternative for development and prototyping stages.

Hidden Costs
- Indirect costs include developer time for configuration, potential differences between local and production environments, and hardware upgrades for resource-intensive applications.

UNIVERSITY *of* WASHINGTON

15



Cost Example – E-Commerce Prototyping

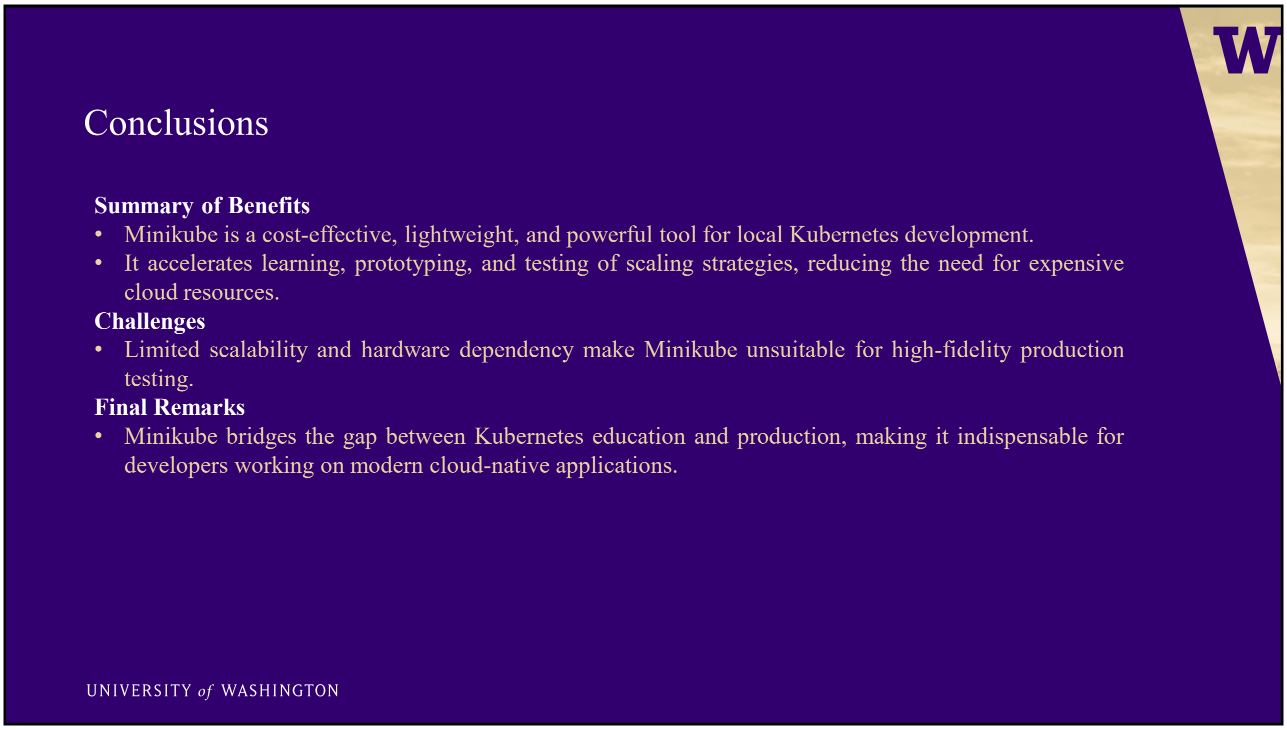
Scenario:
- Testing a Kubernetes-based e-commerce app with Minikube versus cloud.

Minikube:
- No additional cost beyond hardware (e.g., a developer's laptop).
- Ideal for testing scaling strategies and blue-green deployments locally.

Cloud:
- \$50–\$200/month for a small Kubernetes cluster on AWS or GCP (based on node size and load).
- Includes managed services like load balancers and persistent storage.
- Minikube enables cost savings during the prototyping stage before moving to cloud production environments.

UNIVERSITY *of* WASHINGTON

16



Conclusions

Summary of Benefits
- Minikube is a cost-effective, lightweight, and powerful tool for local Kubernetes development.
- It accelerates learning, prototyping, and testing of scaling strategies, reducing the need for expensive cloud resources.

Challenges
- Limited scalability and hardware dependency make Minikube unsuitable for high-fidelity production testing.

Final Remarks
- Minikube bridges the gap between Kubernetes education and production, making it indispensable for developers working on modern cloud-native applications.

UNIVERSITY *of* WASHINGTON

17

Demonstration:

UNIVERSITY *of* WASHINGTON

18