# kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.6.1/components.yaml

Metrics Server often fails due to certificate issues or configuration related to insecure TLS. You may need to adjust its deployment:

**Edit the deployment**:

kubectl edit deployment metrics-server -n kube-system

Locate the container arguments section and add the following parameters under the args:

args:

- --kubelet-insecure-tls

- --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname

These settings bypass TLS verification (suitable for test/lab environments).

A computer screen shot of green and purple text

AI-generated content may be incorrect.

kubectl top nodes

kubectl top pods

A computer screen with green text

AI-generated content may be incorrect.

**✅ Step 3: Simulate Load and Test Autoscaling**

If you'd like to verify the autoscaling, generate load on your application using your provided load-generation script:

* **Run the CPU Load Script inside the Pods**  
  (assuming your container has shell access)

kubectl exec -it <pod-name> -- sh

* Paste your CPU-load script to trigger CPU consumption, or you can create a temporary pod explicitly to generate load:

kubectl run -i --tty load-generator --image=busybox --restart=Never -- /bin/sh -c "while true; do wget -q -O- http://python-deployment; done"

#!/bin/sh

# Function to generate load

generate\_load() {

while true

do

# Perform a computationally intensive task (e.g., calculating Fibonacci numbers)

echo "scale=1000; 4\*a(1)" | bc -l

done

}

# Number of parallel processes to run (adjust based on desired CPU utilization)

num\_processes=10

# Start the specified number of processes

i=0

while [ $i -lt $num\_processes ]

do

generate\_load &

i=$((i + 1))

done

# Display message to indicate the load is being generated

echo "CPU load generation started with $num\_processes processes."

# Wait indefinitely so the script keeps running and the CPU load is sustained

wait