# Seeweb K8S GUIDE

This guide explains how to set up your local development environment to work with an existing Serverless GPU Kubernetes cluster using a provided kubeconfig file. This cover the essential setup of WSL2, Docker, and kubectl.

## Windows Prerequisites Setup

### Required Software Installation

1. Install Docker Desktop from https://www.docker.com/
2. Install WSL 2 following Microsoft's official guide: https://learn.microsoft.com/en-us/windows/wsl/install
3. Restart your system
4. Enable Hyper-V through Windows Features (Run `optionalfeatures.exe` in PowerShell)

### Docker Desktop Configuration

1. Launch Docker Desktop
2. In Settings, ensure:
   - WSL 2 based engine is enabled
   - Resources > WSL Integration is enabled for your Linux distribution

## WSL2 Environment Setup

### Basic Docker Configuration

```
# Verify Docker installation
docker --version

# Remove any old Docker installations
```

```
sudo apt-get remove docker docker-engine docker.io containe
rd runc

# Update package index and install prerequisites
sudo apt-get update
sudo apt-get install \\
    ca-certificates \\
    curl \\
    gnupg \\
    lsb-release

# Add Docker's official GPG key
curl -fsSL <https://download.docker.com/linux/ubuntu/gpg> |
sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-ke
yring.gpg

# Set up the stable repository
echo \\
  "deb [arch=$(dpkg --print-architecture) signed-by=/usr/sh
are/keyrings/docker-archive-keyring.gpg] <https://download.
docker.com/linux/ubuntu> \\
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.li
st.d/docker.list > /dev/null

# Install Docker Engine
sudo apt-get update
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

## Install kubectl

```
# Download latest release
curl -LO "<https://dl.k8s.io/release/$>(curl -L -s <http
s://dl.k8s.io/release/stable.txt>)/bin/linux/amd64/kubectl"

# Install kubectl
sudo install -o root -g root -m 0755 kubectl /usr/local/bi
n/kubectl
```

```
# Verify installation
kubectl version --client
```

### Install Additional Tools

### Kubernetes Management Tools

```
# Install k9s (terminal UI for Kubernetes)
curl -sS <https://webinstall.dev/k9s> | bash

# Install kubectx and kubens for easy context and namespace
switching
sudo git clone <https://github.com/ahmetb/kubectx> /opt/kub
ectx
sudo ln -s /opt/kubectx/kubectx /usr/local/bin/kubectx
sudo ln -s /opt/kubectx/kubens /usr/local/bin/kubens
```

### Shell Configuration

Add these aliases to your `~/.bashrc` to make working with kubectl more convenient:

```
# Kubernetes aliases
echo 'alias k=kubectl' >> ~/.bashrc
echo 'alias kns=kubens' >> ~/.bashrc
echo 'alias kctx=kubectx' >> ~/.bashrc

# Auto-start Docker daemon
echo '# Start Docker automatically
if service docker status 2>&1 | grep -q "is not running"; t
hen
    sudo service docker start
fi' >> ~/.bashrc

source ~/.bashrc
```

## Kubeconfig Setup

The kubeconfig file contains all the necessary information to connect to your Kubernetes cluster, including server addresses, authentication details, and context information. Here's how to set it up properly:

## Setting Up Your Kubeconfig

1. Create the kubectl configuration directory:

```
mkdir -p ~/.kube
```

2. Copy your provided kubeconfig file:

```
# If this is your only cluster
cp /path/to/downloaded/.kubeconfig ~/.kube/config

# OR if you want to keep separate files
cp /path/to/downloaded/.kubeconfig ~/.kube/dev-cluster-c
onfig
```

3. Set proper permissions to protect sensitive information:

```
chmod 600 ~/.kube/config
```

4. Make the configuration permanent by adding to your `.bashrc`:

```
# Add this if you're using multiple config files
echo 'export KUBECONFIG=~/.kube/config:~/.kube/dev-clust
er-config' >> ~/.bashrc
```

## Verify Your Configuration

1. List available contexts:

```
kubectl config get-contexts
```

2. View your current context:

```
kubectl config current-context
```

3. Test your cluster connection:

```
kubectl cluster-info
kubectl get nodes
```

# Container Registry Setup

## GitHub Container Registry Configuration

1. Generate GitHub Personal Access Token:
   - Navigate to GitHub → Settings → Developer settings → Personal access tokens → Tokens (classic)
   - Generate new token with scopes: `write:packages`, `delete:packages`, `repo`
   - Save token securely
2. Set Environment Variables:
   - Windows (PowerShell):

     ```
     [Environment]::SetEnvironmentVariable('GITHUB_TOKEN',
     'your-token-here', 'User')
     ```

   - WSL (add to ~/.bashrc):

     ```
     export GITHUB_TOKEN='your-token-here'
     ```

3. Configure Docker Authentication:

```
echo $GITHUB_TOKEN | docker login ghcr.io -u YOUR_GITHUB
_USERNAME --password-stdin
```

4. Create Kubernetes Secret for Registry:

```
kubectl create secret docker-registry ghcr-secret \\
  --docker-server=ghcr.io \\
  --docker-username=YOUR_GITHUB_USERNAME \\
  --docker-password=$GITHUB_TOKEN
```

# TL;DR

```
# 1. Check Prerequisites
uname -a                    # Check Linux kernel
systemctl status docker     # Check Docker status
echo $PATH                  # Verify path

# 2. Install kubectl
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/k

# Create .kube directory if it doesn't exist
mkdir -p ~/.kube

# Copy config from Windows Desktop
cp /mnt/c/Users/YourUsername/Desktop/kubeconfig ~/.kube/confi

# Set proper permissions
chmod 600 ~/.kube/config

# Add to .bashrc for persistence
echo 'export KUBECONFIG=~/.kube/config' >> ~/.bashrc

# Reload .bashrc
source ~/.bashrc

# 3. Check Cluster/Config
kubectl cluster-info
kubectl config view
kubectl get nodes
kubectl get namespaces

# 4. Basic Kubernetes Commands
kubectl get pods                    # List pods
kubectl get services                # List services
kubectl create -f file.yaml         # Create resource
kubectl apply -f file.yaml          # Apply changes
kubectl delete -f file.yaml         # Delete resource
```

```
kubectl describe pod podname          # Pod details
kubectl logs podname                  # Pod logs
kubectl exec -it podname -- /bin/sh   # Shell into pod
kubectl get deployment                # List deployments
kubectl rollout status deployment     # Deployment status
kubectl get events                    # View cluster events
kubectl top nodes                     # Node resource usage
kubectl api-resources                 # List API resources
```