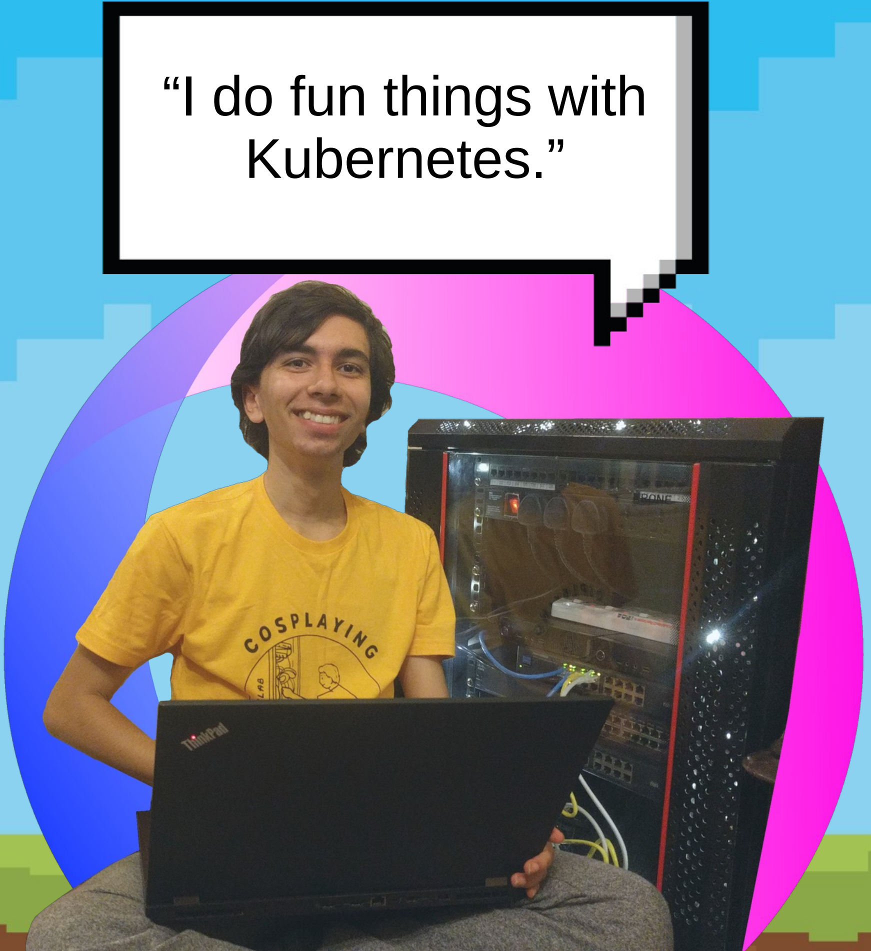
Kubernetes X Frontend MU

By Alex

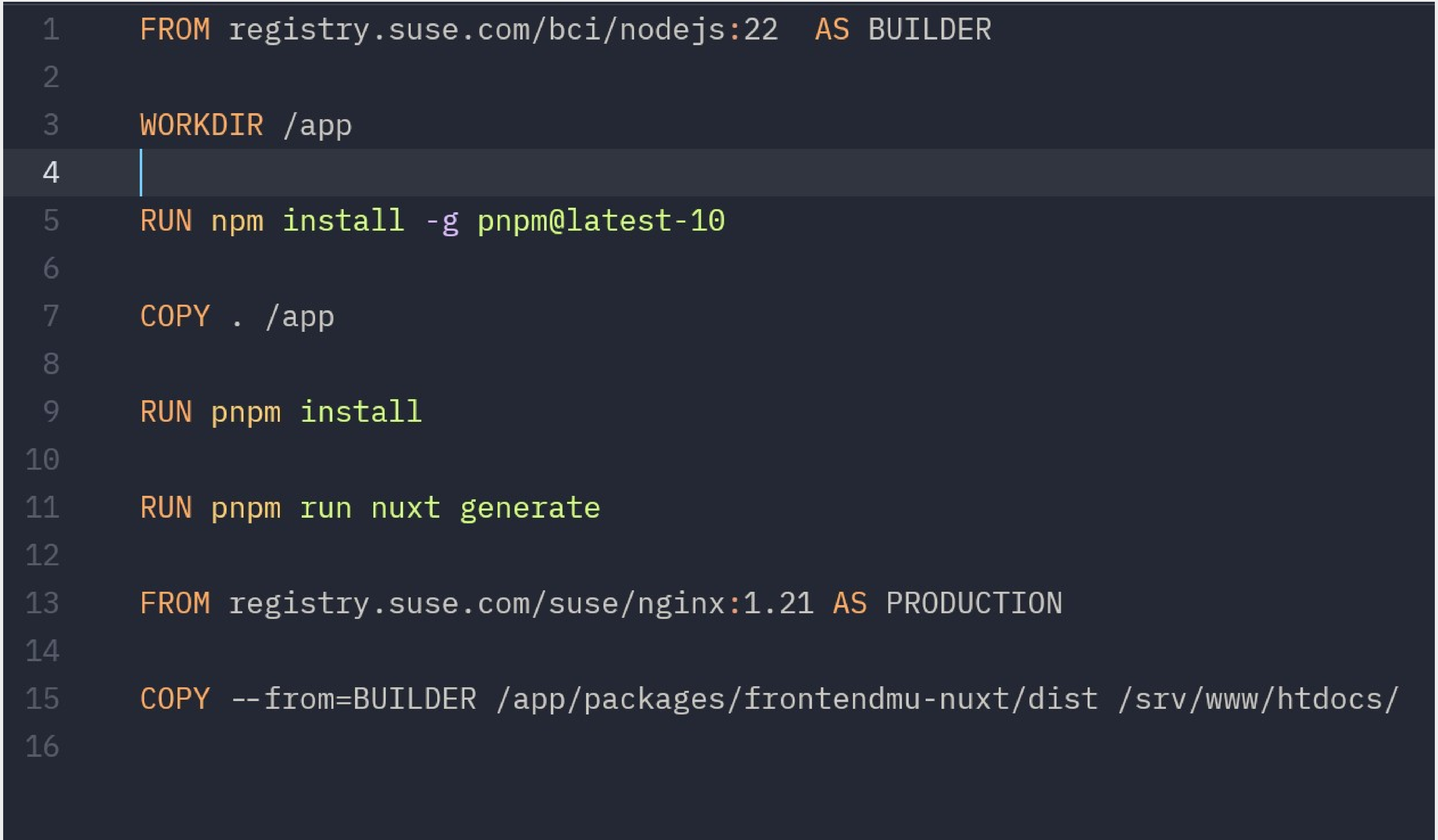
alex.yaml

```yaml
---
apiVersion: v1
kind: Person
metadata:
 name: Alex Bissessur
spec:
  work:
    company: La Sentinelle
    role: Kubernetes Person
    location: Mauritius
  contact:
    website: alexbissessur.dev
    mastodon: moris.social/@AlexB
    github: github.com/xelab04
  interests:
    - Kubernetes
    - Linux
    - Free & Open Source Software
  hobbies:
    - Playing kubectl with Homelab
```



Containers

- Similar to VMs for workload isolation & deployment
- Shares host kernel (Linux) so no need for:
  - kernel
  - hardware emulation
  - OS overhead
- Containers include dependencies & are isolated
- Extremely lightweight (compute & storage)
- For ex: an nginx container with a static site < 50MB

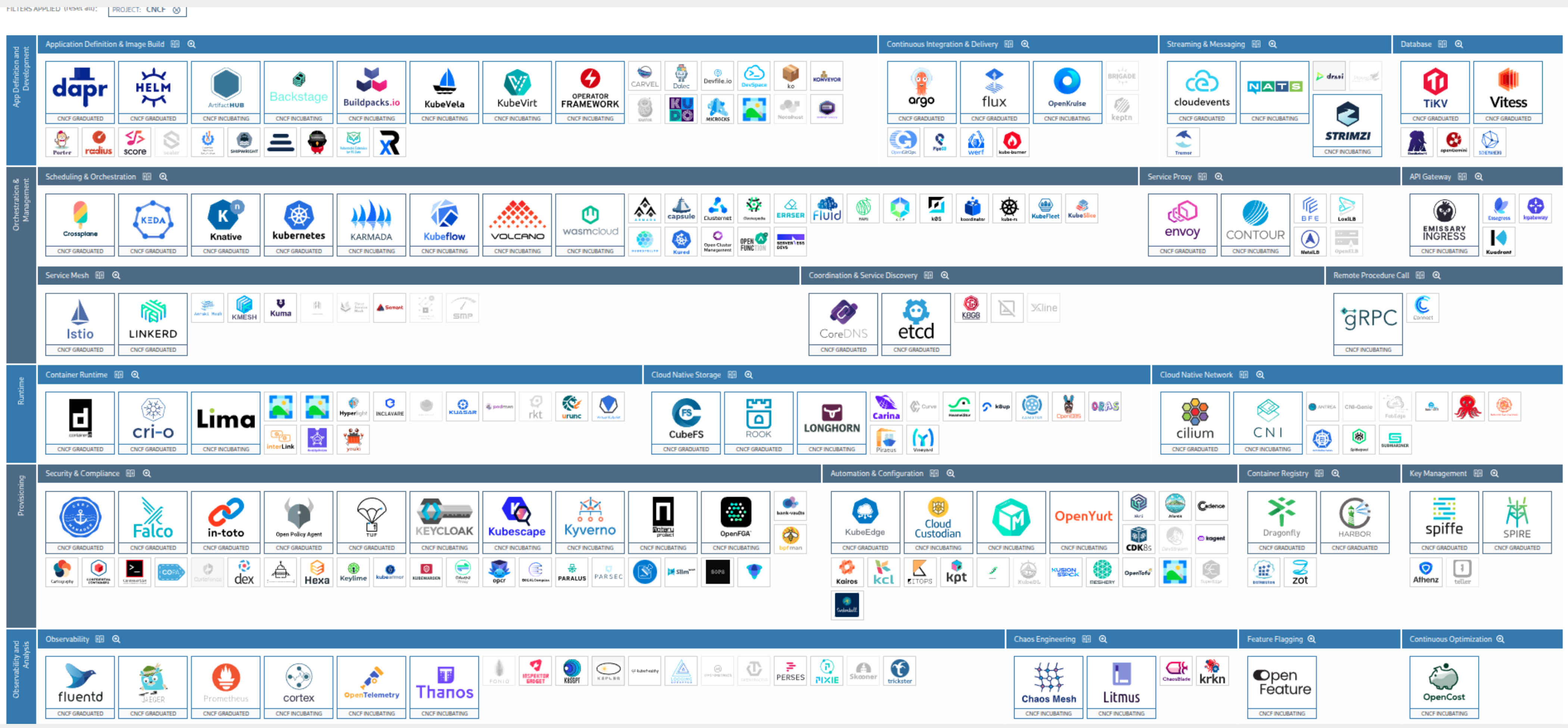


```dockerfile
FROM registry.suse.com/bci/nodejs:22  AS BUILDER

WORKDIR /app

RUN npm install -g pnpm@latest-10

COPY . /app

RUN pnpm install

RUN pnpm run nuxt generate

FROM registry.suse.com/suse/nginx:1.21 AS PRODUCTION

COPY --from=BUILDER /app/packages/frontendmu-nuxt/dist /srv/www/htdocs/

```

Kubernetes

- Industry standard for cluster computing
- “K8S is a system for automating deployment, scaling and management of containerised applications“
  - Simply an abstraction to run a cluster of computers
- Kubernetes in Greek translates to “helmsman” or “pilot” of a ship
- Container orchestrator with a bunch of bonuses

Cloud Native



How Kubernetes Works

- It (intelligently) throws containers onto hosts
- Containers are bundled into pods
  - pods are the smallest unit in Kubernetes
  - pods can have >= 1 container
- It creates networking routes to expose applications
- Gives policies and RBAC and other tools for configuring “stuff”

How To Use Kubernetes

1. Deploy pods to your cluster. This is done with a deployment.

2. Let deployment:
   - keep pods alive
   - update and rollback pods
   - scale pods for high demand

3. Create service and ingress for networking

4. Add autoscaling to handle Black Friday traffic

Why Kubernetes

- Scalability – add more containers / servers
- Redundancy from having several servers
  - high availability
- Maintenance without having downtime
  - blue-green deployments
- Fine-grain control over deployments
  - access perms, resource usage
- Easy CI/CD integrations

Why Kubernetes

- Self-healing – no need to call sysadmin at 3am when server dies
- Portability – stateless containers can be moved to different clusters easily (eg test env)
- Cloud-agnostic (also good for bare-metal)
- Easy to manage – less workload on sysadmins for server management
- Automatic pod scaling to match demand

Why Not Kubernetes?

- For simplicity
  - K8s needs someone who knows about infra
  - Simple docker compose on $5 VPS works
- Learning Curve
  - Devs (esp frontend ppl) are not familiar with this approach to infra
- Legacy (fat) Projects
  - Not easy to containerise

Demo Time