Simplifying your life with Docker, Jenkins and Minikube

Juan Peredo
linkedin.com/in/juanperedotech

Another day at the office

Another day at the office

- Boss stops by your desk

Another day at the office

- Boss stops by your desk
- Customer wants a "small" change in application

Another day at the office

- Boss stops by your desk
- Customer wants a "small" change in application
  - It was requested two weeks ago

Another day at the office

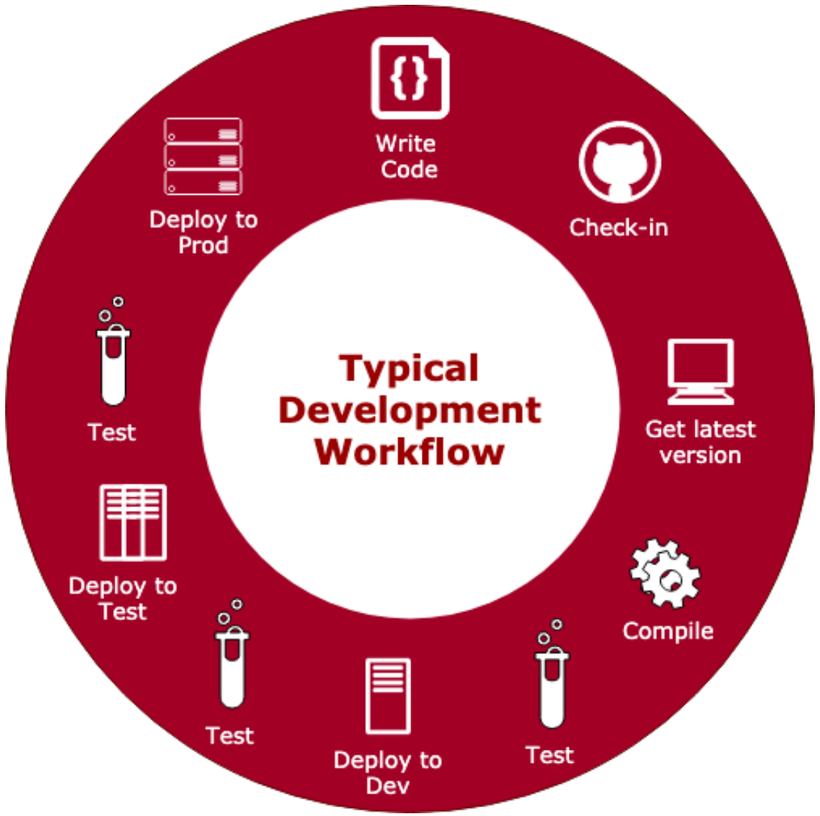
- Boss stops by your desk
- Customer wants a "small" change in application
  - It was requested two weeks ago
  - They forgot to tell you

Another day at the office

- Boss stops by your desk
- Customer wants a "small" change in application
  - It was requested two weeks ago
  - They forgot to tell you
  - It was due yesterday

How you really feel

Long change cycle

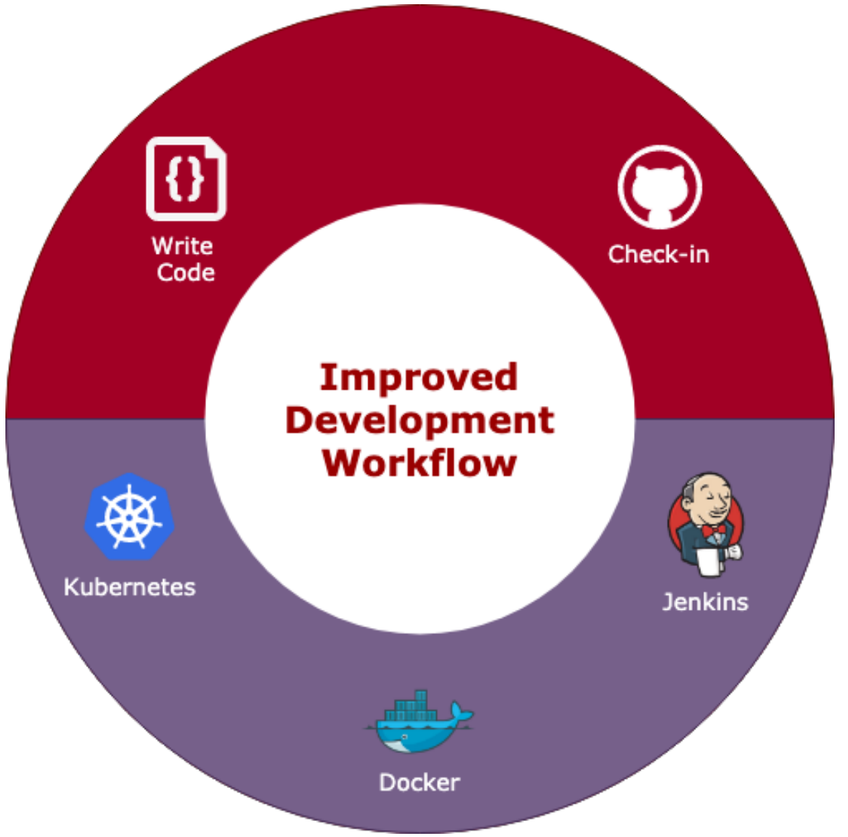


- Involve a bunch of manual steps
- Most of the steps do not add value
- Time consuming and waste resources
  - Up to 40% of a regular day is spent on non value add tasks (environment setup, waiting for tests & builds, etc)[1].
- A single missed step can compromise the whole application

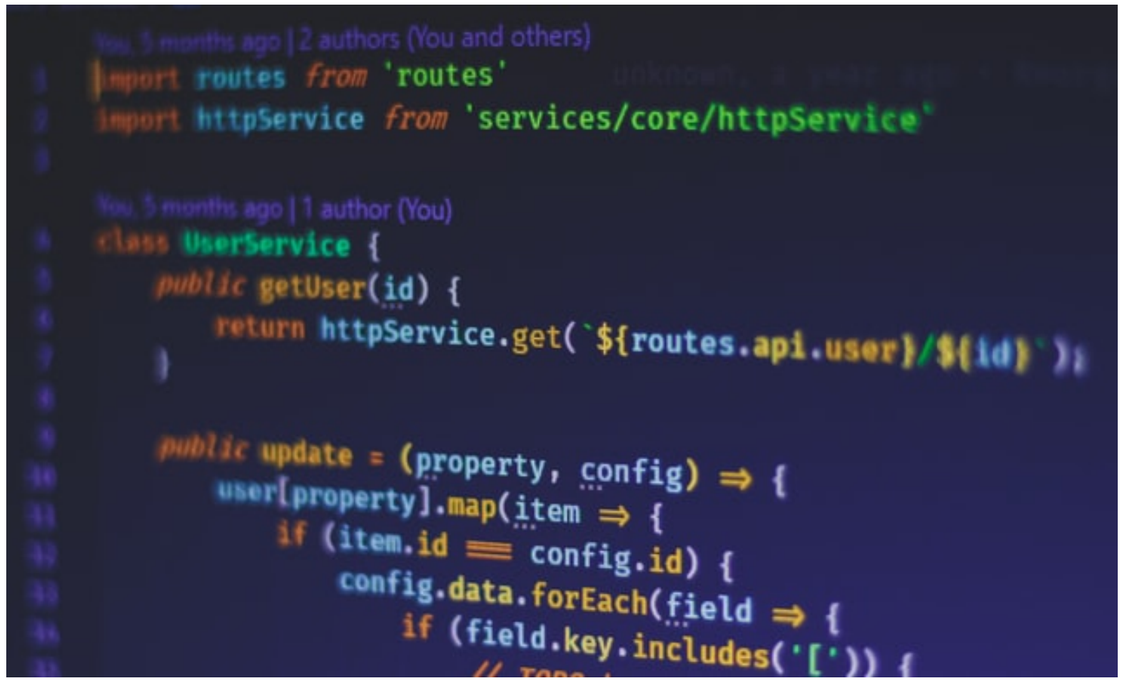
[1] infoworld

It does not have to be like this!

- There are tools that can help to automate the process. For example:
  - Containers simplify the bundling of the application
  - Tools like Jenkins, TravisCI and Atlassian Pipelines help with CI/CD
  - Kubernetes manages the orchestration
- The goal is to allow developers to focus on value-add tasks!



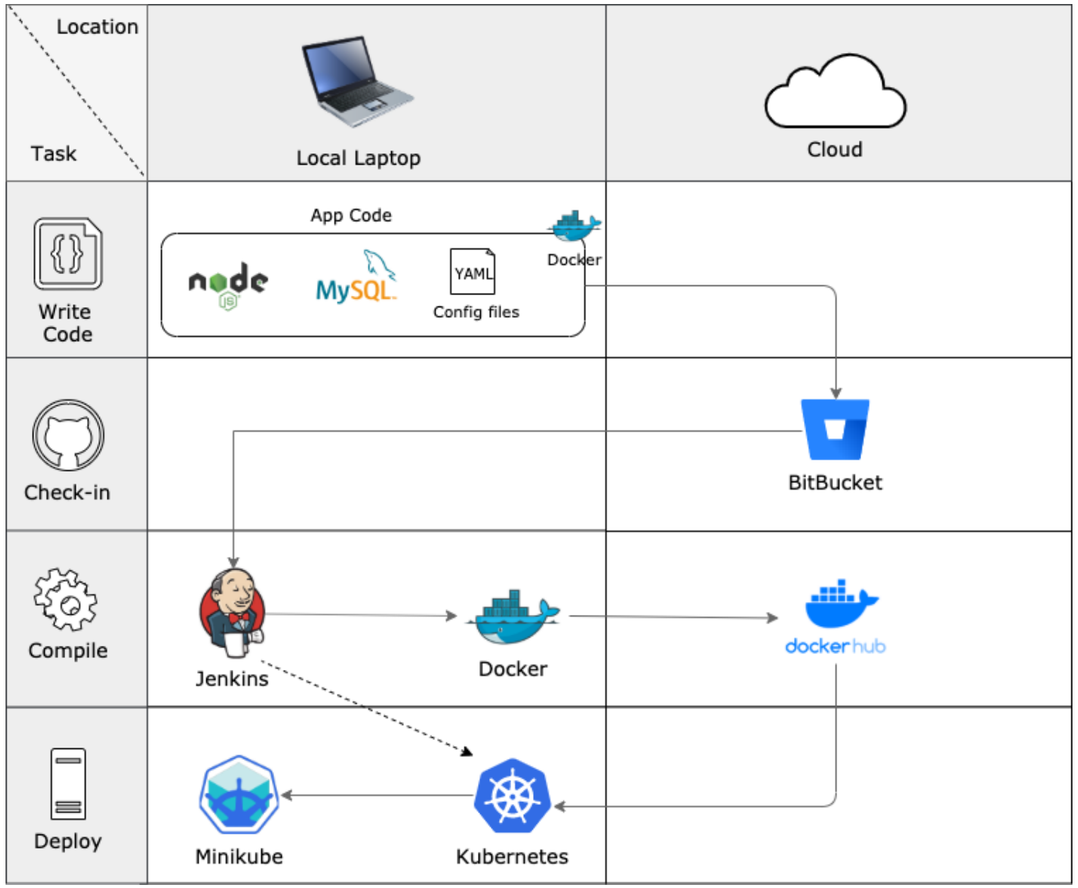
Demo Time



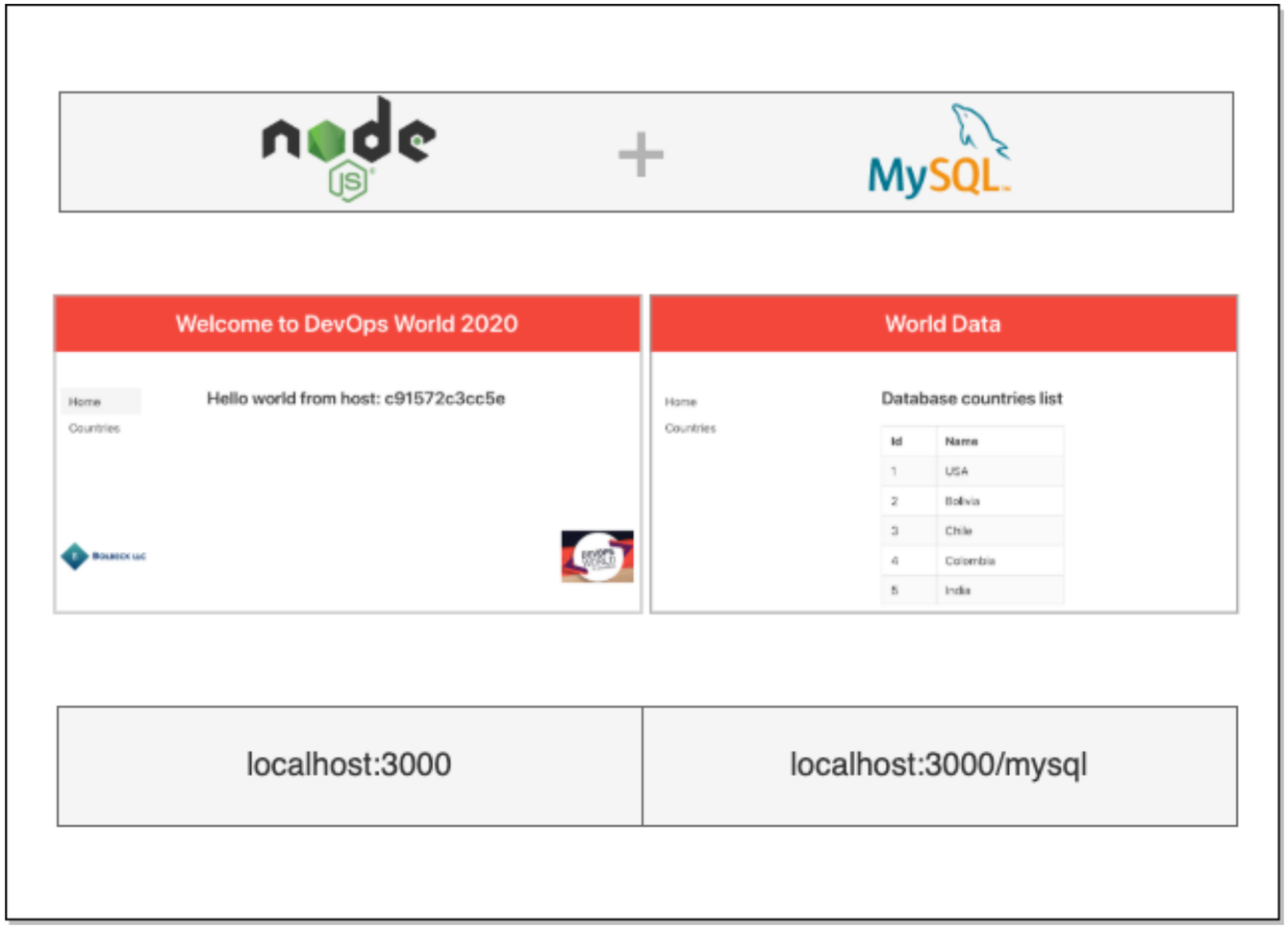
Goals

- Automate typical every day tasks so that we can focus on creating awesomeness!
- Do everything (or most) in code so that it is repeatable & source controlled

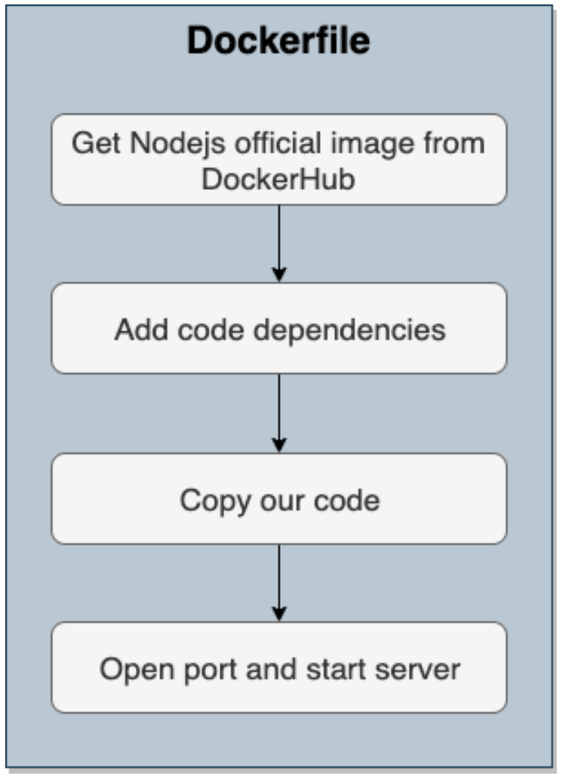
Demo Roadmap



Our Application

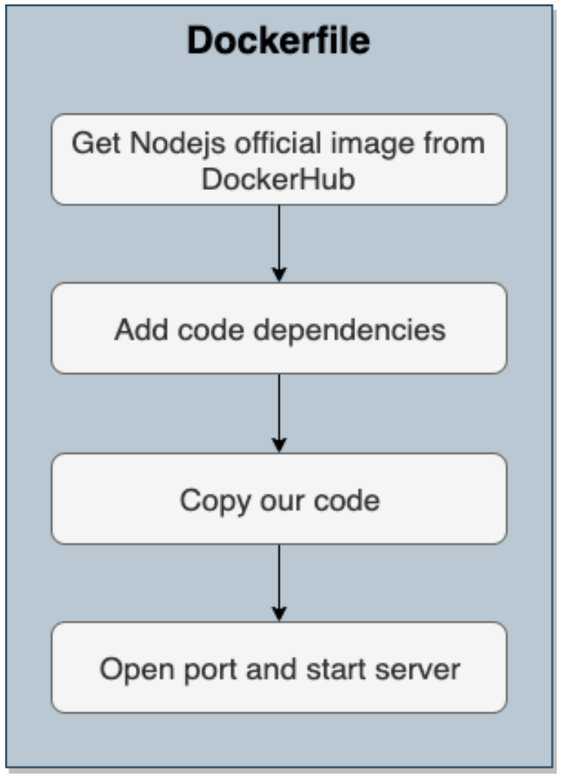


Custom Nodejs Image



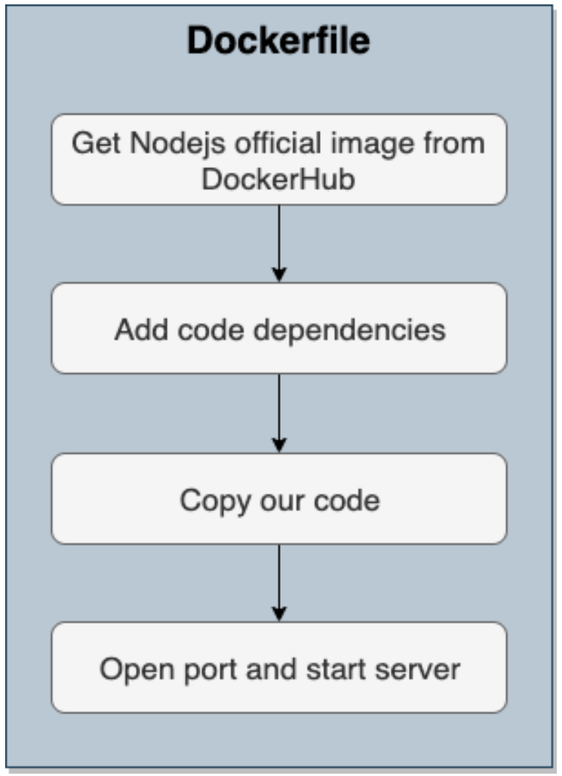

```
FROM node
WORKDIR /code
COPY package.json /code
RUN npm install mysql2
RUN npm install express
RUN npm install chai
RUN npm install chai-http
RUN npm install mocha
RUN npm install mocha-junit-reporter

COPY . /code
RUN npm install

EXPOSE 3000
CMD ["npm","start"]
```

Custom Nodejs Image



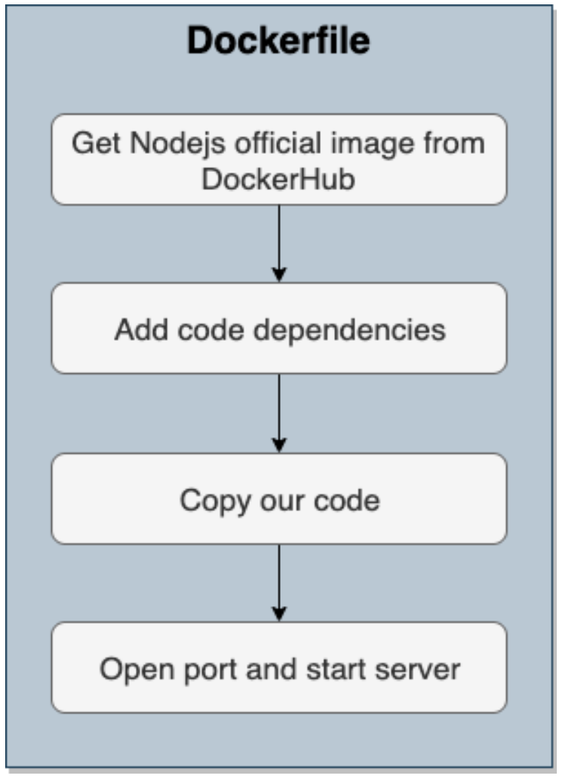

```
FROM node
WORKDIR /code
COPY package.json /code
RUN npm install mysql2
RUN npm install express
RUN npm install chai
RUN npm install chai-http
RUN npm install mocha
RUN npm install mocha-junit-reporter

COPY . /code
RUN npm install

EXPOSE 3000
CMD ["npm","start"]
```

Custom Nodejs Image



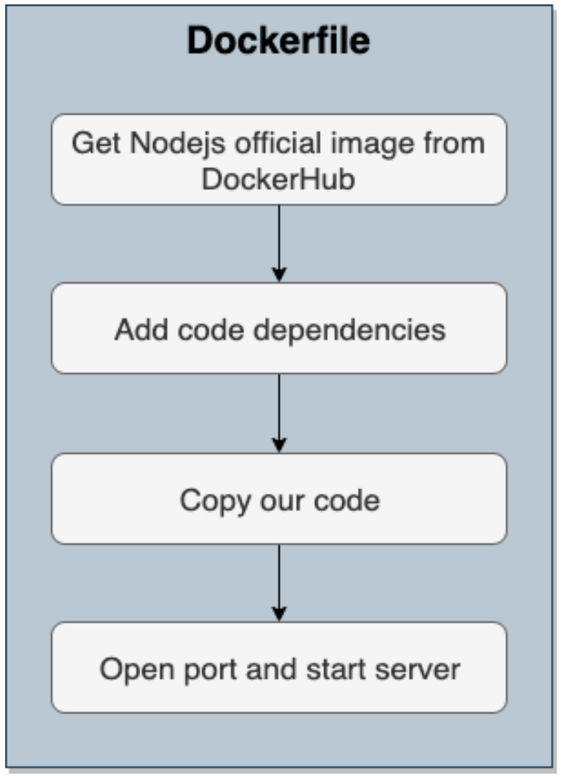

```
FROM node
WORKDIR /code
COPY package.json /code
RUN npm install mysql2
RUN npm install express
RUN npm install chai
RUN npm install chai-http
RUN npm install mocha
RUN npm install mocha-junit-reporter

COPY . /code
RUN npm install

EXPOSE 3000
CMD ["npm","start"]
```

Custom Nodejs Image



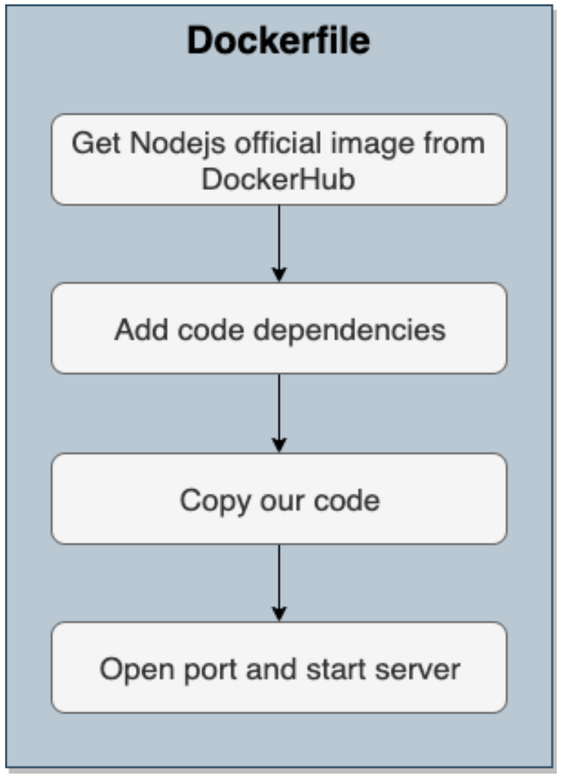

```
FROM node
WORKDIR /code
COPY package.json /code
RUN npm install mysql2
RUN npm install express
RUN npm install chai
RUN npm install chai-http
RUN npm install mocha
RUN npm install mocha-junit-reporter

COPY . /code
RUN npm install

EXPOSE 3000
CMD ["npm","start"]
```

Custom Nodejs Image



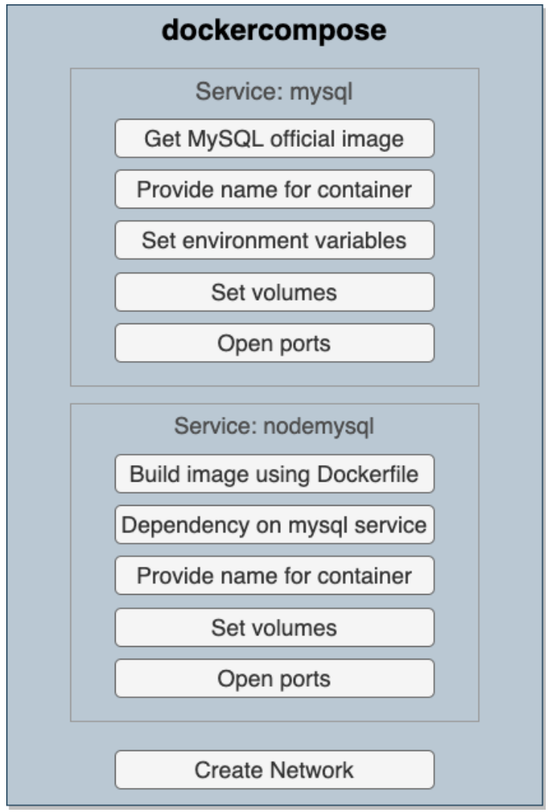

```
FROM node
WORKDIR /code
COPY package.json /code
RUN npm install mysql2
RUN npm install express
RUN npm install chai
RUN npm install chai-http
RUN npm install mocha
RUN npm install mocha-junit-reporter

COPY . /code
RUN npm install

EXPOSE 3000
CMD ["npm","start"]
```

Custom Nodejs Image



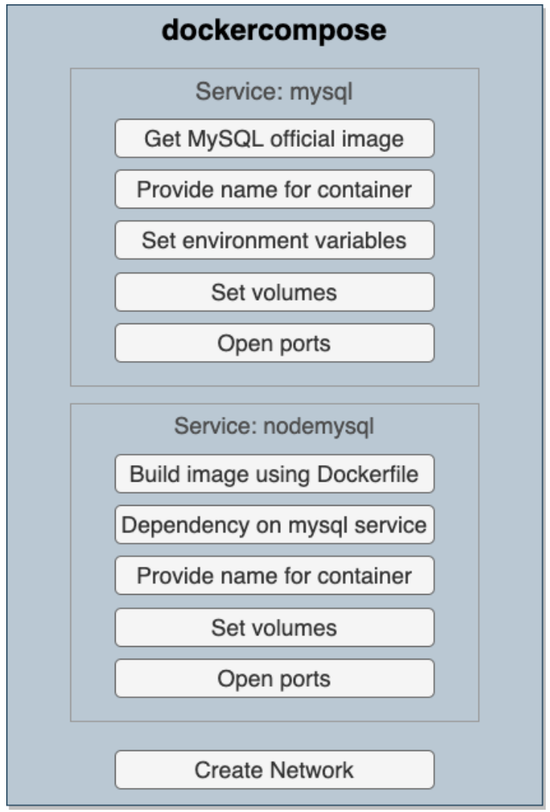

```
FROM node
WORKDIR /code
COPY package.json /code
RUN npm install mysql2
RUN npm install express
RUN npm install chai
RUN npm install chai-http
RUN npm install mocha
RUN npm install mocha-junit-reporter

COPY . /code
RUN npm install

EXPOSE 3000
CMD ["npm","start"]
```

Nodejs connected to MySQL



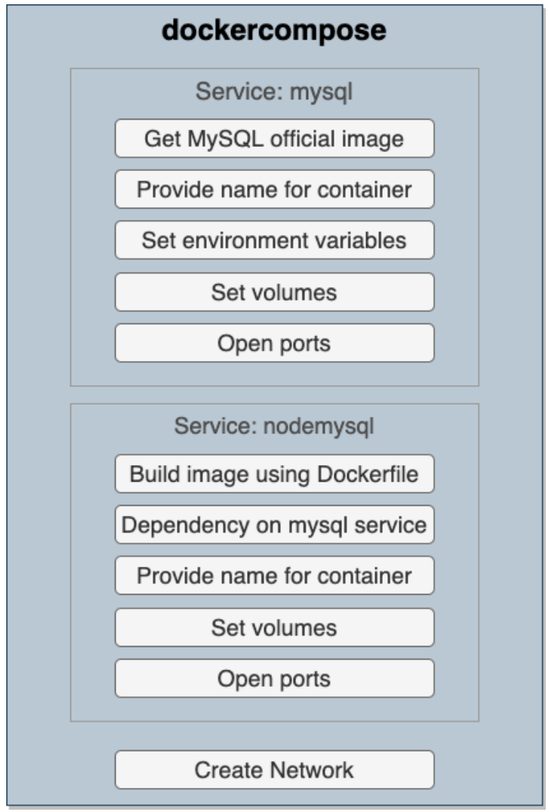

```
version: "3.7"
services:
  mysql2:
    image: mysql
    container_name: mysql2
    env_file: docker-compose.env
    volumes:
      - ./mySqlDB:/var/lib/mysql
      - ./mySqlInit:/docker-entrypoint-initdb.d
    ports:
      - "3306:3306"
  nodemysql:
    build: ./nodeApp
    depends_on:
      - mysql2
    container_name: nodemysqlcont
    volumes:
      - ./nodeApp:/code
    ports:
      - "3000:3000"
```

Nodejs connected to MySQL



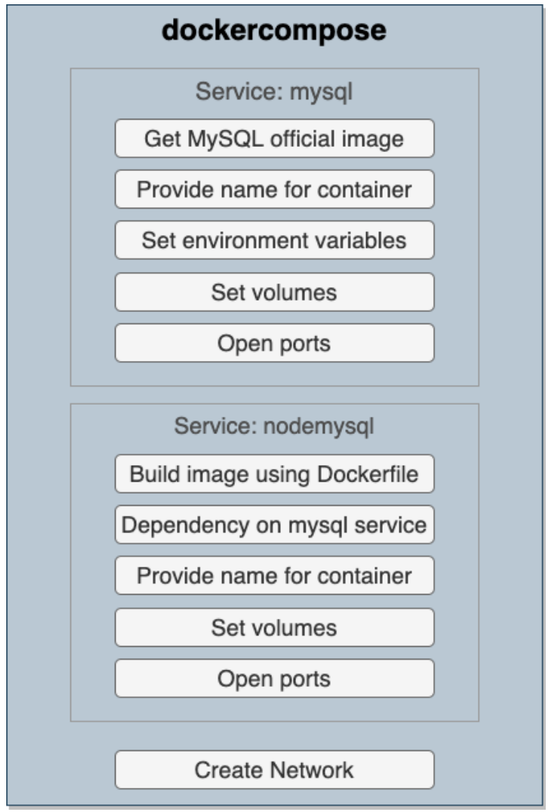

```yaml
version: "3.7"
services:
  mysql2:
    image: mysql
    container_name: mysql2
    env_file: docker-compose.env
    volumes:
      - ./mySqlDB:/var/lib/mysql
      - ./mySqlInit:/docker-entrypoint-initdb.d
    ports:
      - "3306:3306"
  nodemysql:
    build: ./nodeApp
    depends_on:
      - mysql2
    container_name: nodemysqlcont
    volumes:
      - ./nodeApp:/code
    ports:
      - "3000:3000"
```

Nodejs connected to MySQL




```
version: "3.7"
services:
  mysql2:
    image: mysql
    container_name: mysql2
    env_file: docker-compose.env
    volumes:
      - ./mySqlDB:/var/lib/mysql
      - ./mySqlInit:/docker-entrypoint-initdb.d
    ports:
      - "3306:3306"
  nodemysql:
    build: ./nodeApp
    depends_on:
      - mysql2
    container_name: nodemysqlcont
    volumes:
      - ./nodeApp:/code
    ports:
      - "3000:3000"
```

Nodejs connected to MySQL



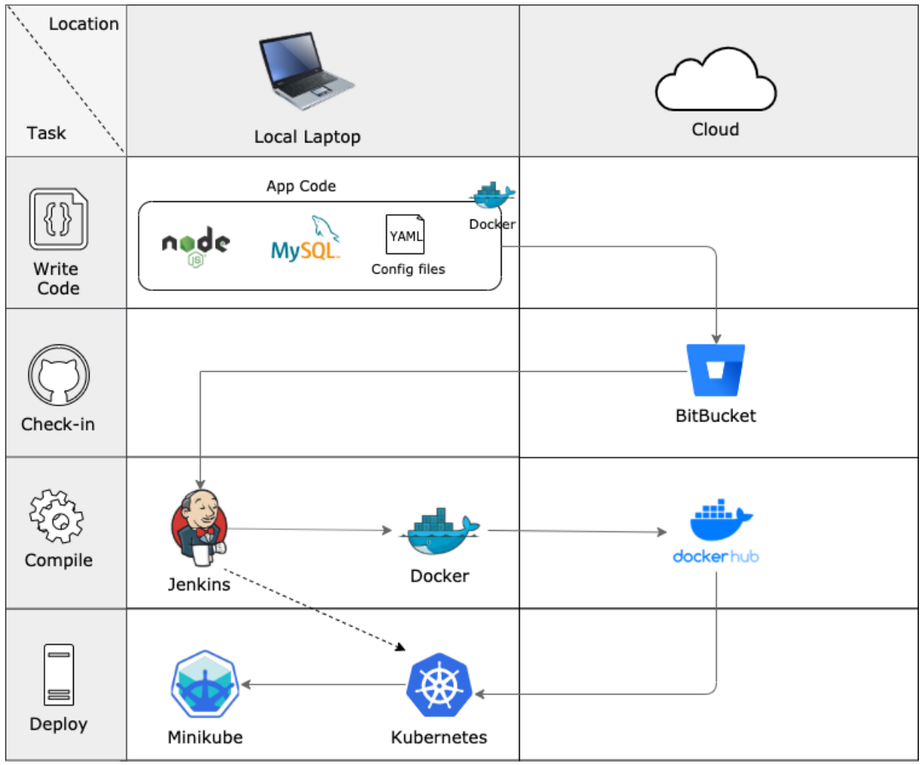

```
version: "3.7"
services:
  mysql2:
    image: mysql
    container_name: mysql2
    env_file: docker-compose.env
    volumes:
      - ./mySqlDB:/var/lib/mysql
      - ./mySqlInit:/docker-entrypoint-initdb.d
    ports:
      - "3306:3306"
  nodemysql:
    build: ./nodeApp
    depends_on:
      - mysql2
    container_name: nodemysqlcont
    volumes:
      - ./nodeApp:/code
    ports:
      - "3000:3000"
```

Dockercompose awesomeness

- Dockercompose contains everything needed for our app to run
- Brings up our containers & creates the appropriate network
- Can be checked in with the rest of our code

docker-compose file

Demo Roadmap

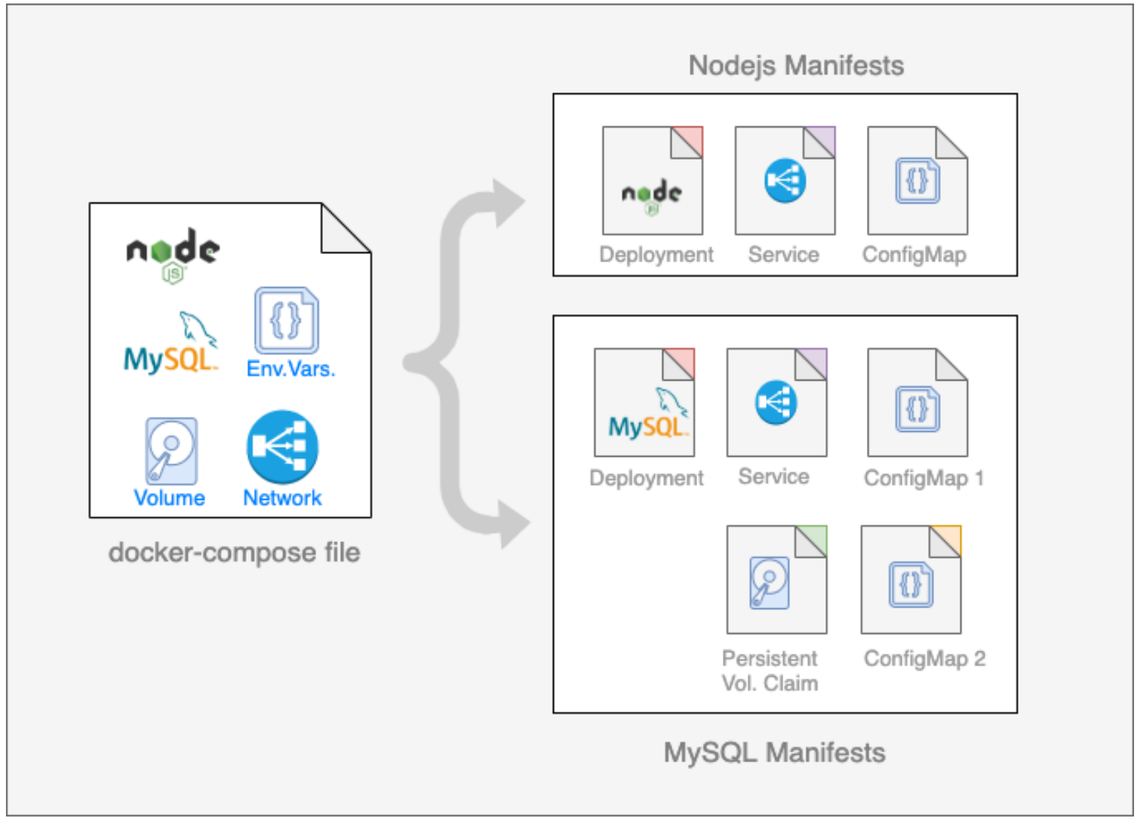


✔ Create docker images for our application

- Check in code to Bitbucket
- Use Jenkins to automate image build and deployment

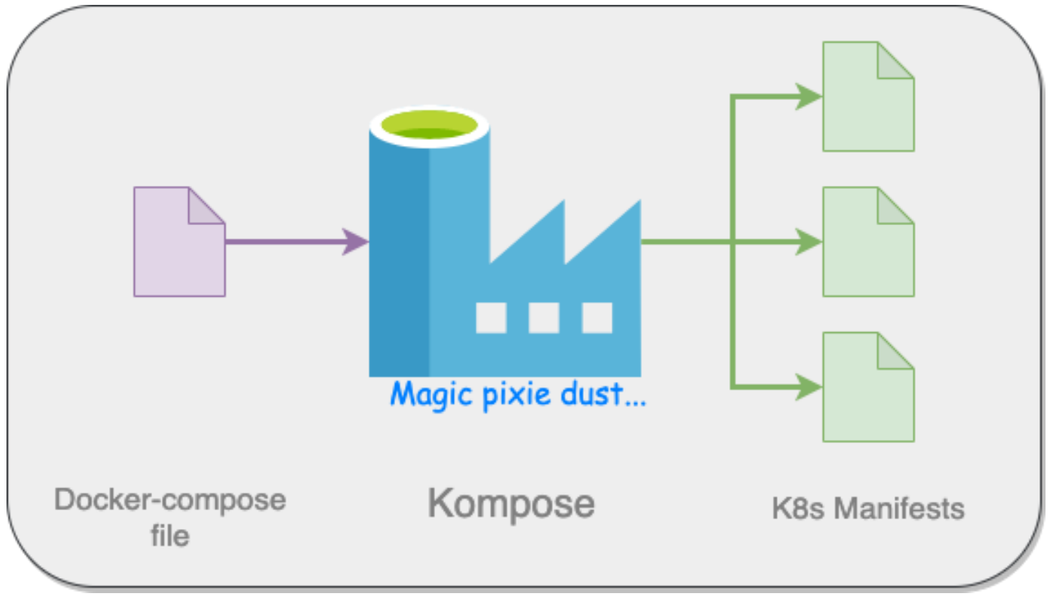
➡ Deploy to Kubernetes in Minikube

Kubernetes Manifests



- With Docker-compose we fully defined our application with two services (Nodejs and MySQL)
- For Kubernetes, we need to map those to multiple resource manifests:
  - Deployments
  - Services
  - Persistent Volumes
  - ConfigMaps

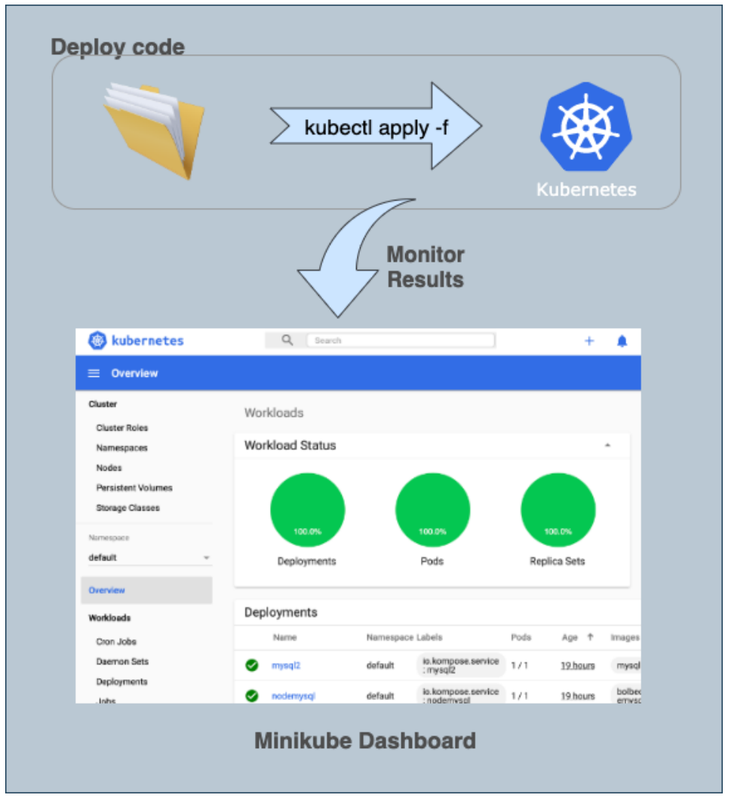
Creating Kubernetes Manifests



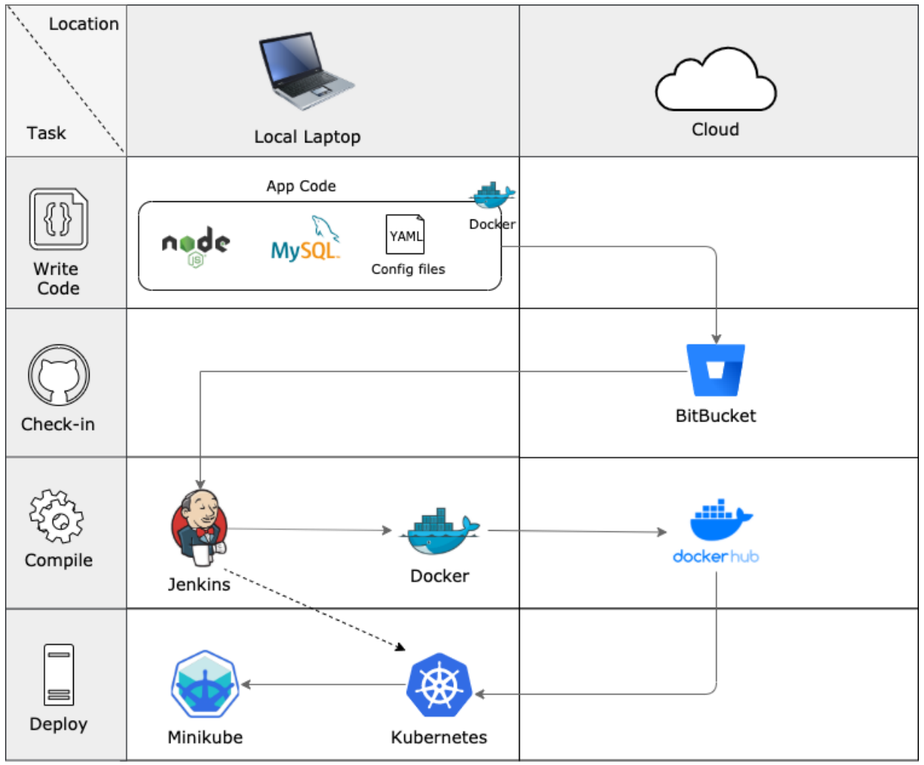
- Kompose generates manifests automagically! 😎
  - Based on dockercompose file
  - Out of the box, generation gets you 90% there
  - Run: `kompose -f <path_to_dokercompose_file> convert`
  - Get Kompose at kompose.io

Deploy to Kubernetes

- Use `Kubectl apply -f <filename or folder name>` to deploy our manifests
- Monitor cluster in the Minikube dashboard
- Or, use `kubectl get all` to see cluster resources
- Delete resources with `Kubectl delete -f <filename or folder name>`



Demo Roadmap



✔ Create docker images for our application

➡ Check in code to Bitbucket

➡ Use Jenkins to automate image build and deployment

✔ Deploy to Kubernetes in Minikube

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                  nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Automating with Jenkins


```
node {def app
  stage('Clone repository') { checkout scm }
  stage('Build image') {
    dir('nodeApp') {
      app = docker.build("bolbeck/simplenodemysql")}}
  stage('Test image') {
    app.withRun{ c ->
      sh "docker exec  ${c.id} npm install"
      sh "docker exec  ${c.id} npm run test-exp"
      sh "docker cp ${c.id}:/code/test-results.xml
                nodeApp/test/test-results.xml" }}
  stage('Publish test results') {
    junit 'nodeApp/test/results/test-results.xml'}
  stage('Push image') {
    docker.withRegistry('https://registry.hub.docker.com',
            'docker-hub-credentials') {
          app.push("${env.BUILD_NUMBER}")
          app.push("latest") }}}
   stage('Deploy to K8s') {
     sh "kubectl apply -f ./Kubernetes/" }
```

Kubectl on Jenkins image

Jenkins image does not come with kubectl, so we create an image that contains both

```
FROM jenkinsci/blueocean

USER root
RUN curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl
RUN chmod u+x kubectl && mv kubectl /bin/kubectl

COPY ./kubeconfig /root/.kube
COPY ./minikConfig  /root/.minikube
```

Kubectl on Jenkins image

Jenkins image does not come with kubectl, so we create an image that contains both

```
FROM jenkinsci/blueocean

USER root
RUN curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl
RUN chmod u+x kubectl && mv kubectl /bin/kubectl

COPY ./kubeconfig /root/.kube
COPY ./minikConfig  /root/.minikube
```

Kubectl on Jenkins image

Jenkins image does not come with kubectl, so we create an image that contains both

```
FROM jenkinsci/blueocean

USER root
RUN curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl
RUN chmod u+x kubectl && mv kubectl /bin/kubectl

COPY ./kubeconfig /root/.kube
COPY ./minikConfig  /root/.minikube
```

Kubectl on Jenkins image

Jenkins image does not come with kubectl, so we create an image that contains both

```
FROM jenkinsci/blueocean

USER root
RUN curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl
RUN chmod u+x kubectl && mv kubectl /bin/kubectl

COPY ./kubeconfig /root/.kube
COPY ./minikConfig  /root/.minikube
```

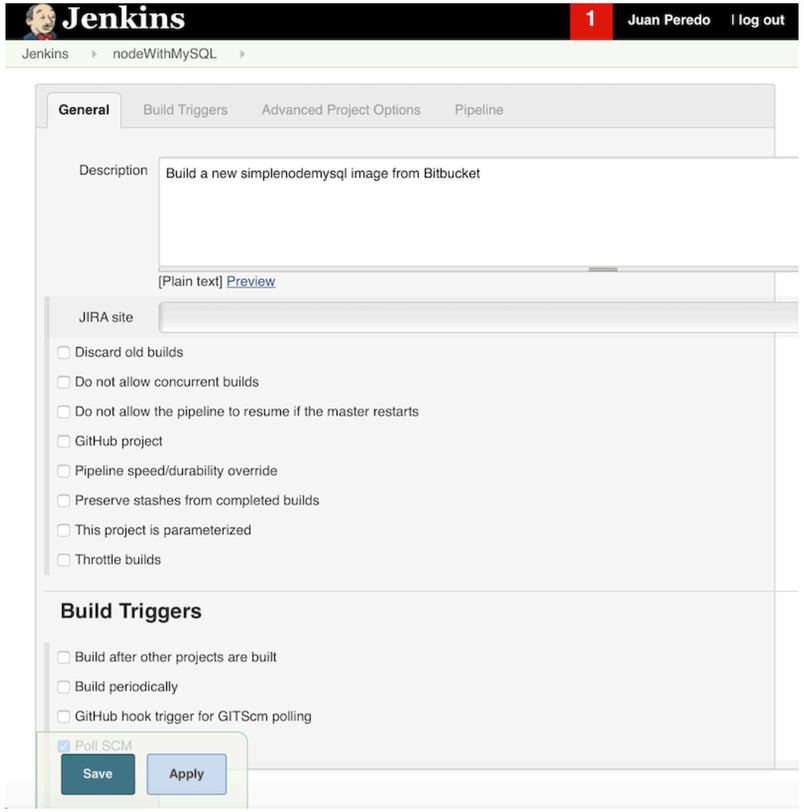
Kubectl on Jenkins image

Jenkins image does not come with kubectl, so we create an image that contains both

```
FROM jenkinsci/blueocean

USER root
RUN curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.16.2/bin/linux/amd64/kubectl
RUN chmod u+x kubectl && mv kubectl /bin/kubectl

COPY ./kubeconfig /root/.kube
COPY ./minikConfig  /root/.minikube
```

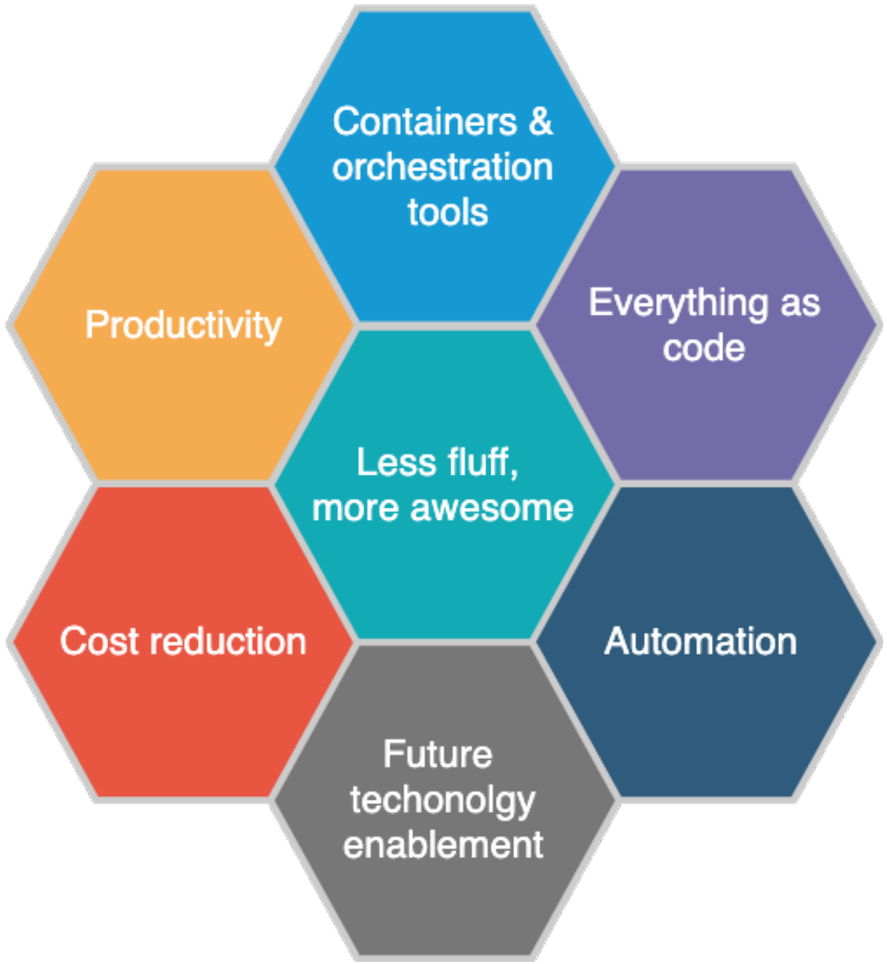
Register pipeline in Jenkins UI

- Add Docker Hub and Bitbucket credentials to Jenkins
- Create a new pipeline Job



Putting it all together...

Key take aways



More importantly …

We can spend more time with the ones we love

Questions ?

Thanks to CloudBees & all the conference sponsors

Simplifying your life with Docker, Jenkins and Minikube

Juan Peredo
linkedin.com/in/juanperedotech
@JuanPeredoTech

Appendix

Photos

-  Photo by Ferenc Almasi on Unsplash
-  Photo by Ben White on Unsplash
-  Photo by Robin Higgins from Pixabay
-  Photo by Robin Higgins from Pixabay
-  Photo by Joshua Clay on Unsplash