Rapid Deployment For Hackathons

Introduction to Hackathon DevOps

Oh no, there is 60 min left to get this server deployed or this this project wont work. Why does this always happen! aaaaaaahhhhhhhhhhhh!

Follow Along:

https://github.com/eyesniper2/RapidDeploymentForHackathonsWorkshop

Outcome

To give you a toolbox of resources to use at hackathons and a glimpse at what real world DevOps look like.

Outcome

This talk will not cover all best possible DevOps practices.

Prerequisites

- Basics of Git (Switch branches/push/pull)
- You know how to/or have made, something that needs to be deployed

Disclaimer

These thoughts, recommendations and opinions are mine and do not reflect the views or practices of employer's past or present.

What is DevOps?

DevOps

DevOps is a set of practices that combines software **development** (*Dev*) and **information-technology operations** (*Ops*) which aims to shorten the systems development life cycle and provide continuous delivery with high software quality.

Source: https://en.wikipedia.org/wiki/DevOps

Goals in DevOps

1. Automated predictable deployments
2. Automated Rollbacks
3. Automated Testing
4. Health Monitoring
5. Ability to Scale
6. Continuous Integration
7. Continuous Deployment

Goals in Hackathon DevOps

1. Get it running for the demo

Services Building Blocks

For this talk we will focus on two pieces:

Databases

- For reading and writing large amounts of data
- SQL (Such as MySQL, Postgres), NoSQL (such as Mongo), Flat File (S3)

database

Virtual Machines/Servers

- The “computer” that runs your services/program
- Come in various specs.

server

High Level DevOps Security Checklist

Security – Part 1

Lockdown ports

For a web server you typically only need 80 open. You should be able to whitelist just your IP address to access the SSH port.

Do NOT hardcode secrets into your codebase

Instead use enviromental variables.

Secure your Database

Only people who should be able to connect to the database can.
Avoid allowing public network access to your database.

Security – Part 2

Do NOT trust your client

Always check user permissions/authentication before allowing access to data on your server.

Use HTTPS, not HTTP

Use tools such as CertBot to get a free certificate.

Be critical when taking advice

Make sure you understand what will happen and why it will happen when taking the advice of others.

Some tools I like

Tools

Postman

API Development Tools + API Server Mocking Tools

Chrome Developer Tools

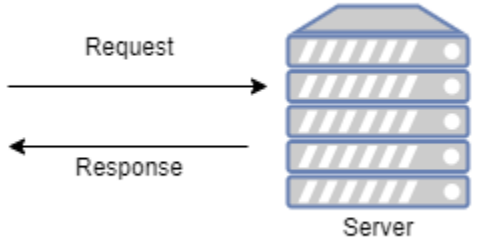
Great for general web development tools.

Fiddler (Advanced)

Web Debugging Proxy. Will let you log and modify web traffic in and out of your machines.

Hackathon DevOps Cookbook

Single Page WebApp



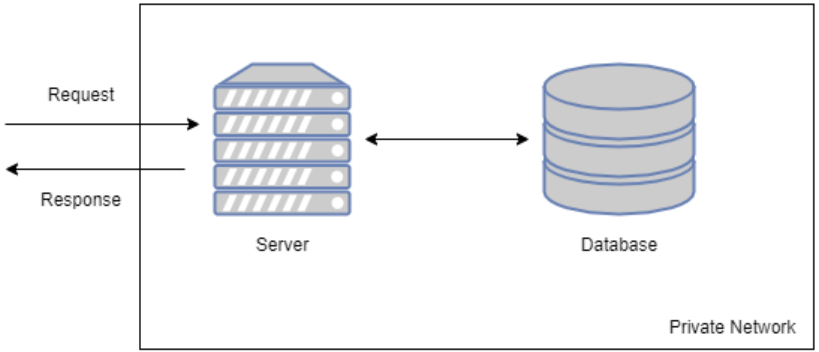
Characteristics

- No database
- Examples include Create a React app. The HackED Website is an example

Tool to use:

- GitHub Pages

API Service / App Backend



Characteristics

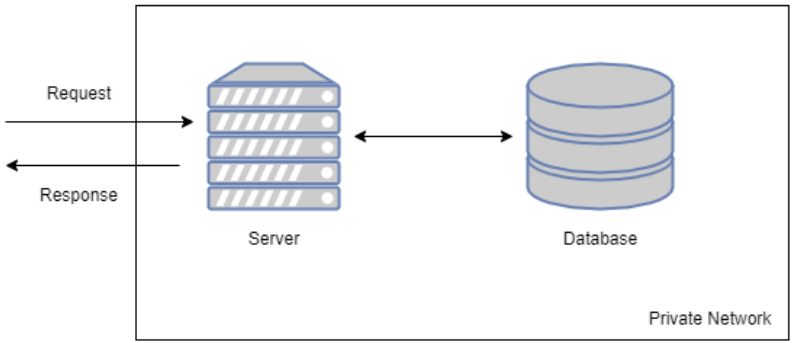
- Handles REST calls
- Example tech stacks: Node.js, flask, .NET

Tool to use:

- Heroku
- ngrok
- Firebase

You can also mock your API using Postman: https://learning.getpostman.com/docs/postman/mock-servers/intro-to-mock-servers/

Full Stack WebApp



Characteristics

- Handles REST calls along with serving up website content
- Example tech stacks: Node.js, Django, flask, .NET

Tool to use:

- ngrok
- Heroku
- Firebase

Let’s Deploy Some Apps!

1. Single Page WebApp

React + GitHub Pages

Steps

1. Have a working React App + GitHub Repo

2. Install the gh-pages dev-dependency

Via: `npm install gh-pages --save-dev`

3. Add/Edit the homepage property in package.json

Add/edit the home page property to look like: `"homepage": "http://{username}.github.io/{repo-name}"`

Steps

4. Add deploy step in Package.json scripts

```
"predeploy": "npm run build",
"deploy": "gh-pages -d build"
```

5. Build then deploy

```
Run: npm run build
Then run: npm run deploy
```

Reference: https://dev.to/yuribenjamin/how-to-deploy-react-app-in-github-pages-2a1f

2. API Service

Node.JS + ngrok

Steps

1. Have a somewhat working web server

This time you don’t need a GitHub Repo

2. Download ngrok and create an account (free)

Via: https://ngrok.com/download

3. Connect your account

Using a terminal run the command:

```
ngrok authtoken <YOUR_AUTH_TOKEN>
```

Steps

4. Start your local dev server

Start your local server. Take note of what port your local server is running on.

5. Start ngrok

Using a terminal run the command:

```
ngrok http <Your_Local_Server_Port>
```

6. Start making requests to your website!

3. Full Stack WebApp

Node.JS + React + Heroku

Steps

1. Have your project in a GitHub Repo

This time you will need a GitHub Repo

2. Install the Heroku CLI and create an account

https://devcenter.heroku.com/articles/heroku-cli

3. Login to your Heroku account

```
heroku login
```

Steps

4. Create your Heroku App

```
heroku create
```

5. (Optional) Create your Database

```
heroku addons:create heroku-postgresql:hobby-dev
```

6. Push your app

```
git push heroku master
```

Steps

4. (Optional) Connect to your database

```
heroku pg:psql
```

Other useful commands:

```
heroku logs --tail
heroku run bash
```

Yay! In an hour we have deployed 3 different Web Apps

Thanks for coming!

Any Questions?

Presentation and Examples:

https://github.com/eyesniper2/RapidDeploymentForHackathonsWorkshop