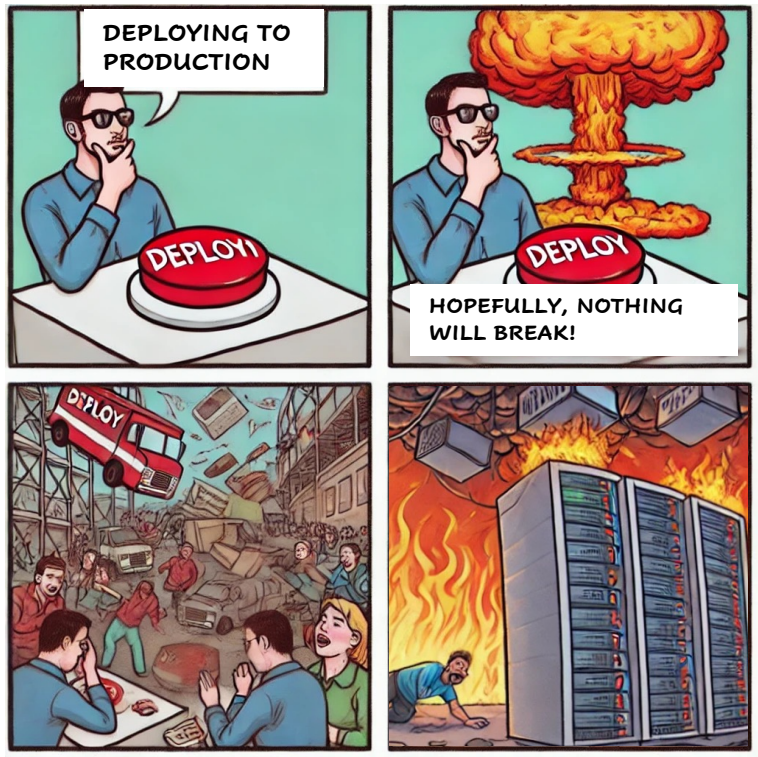


Deployment

CSC309

Kianoosh Abbasi

UNIVERSITY OF TORONTO

Fall 2024

This session

- Hosting your application
- Backend and frontend deployment
- DevOps

What is a server?

- It's another computer (or virtual machine), like your local one!
- It should have a static public IP address
  - And maybe a domain bound to that IP
- Get a server from AWS, Azure, Google Cloud, etc.
  - Some platforms offer small virtual machines (VM) for free!
- But that's not only about IP and the machine!

Get a domain

< Step 2: DNS and Host Records options

3/3

DNS / Host Records

Host Records options

What to do with Host Records?

Any changes to host records and email settings here will impact all selected domains. Due to propagation, changes will take some time to apply. Follow this link for more info.Learn More →

Click to add Host Records

A | AAAA | CNAME | NS | SPF | TXT | URL REDIRECT

A RECORD

frontend	@	26.78.129.90	30 minutes

A RECORD

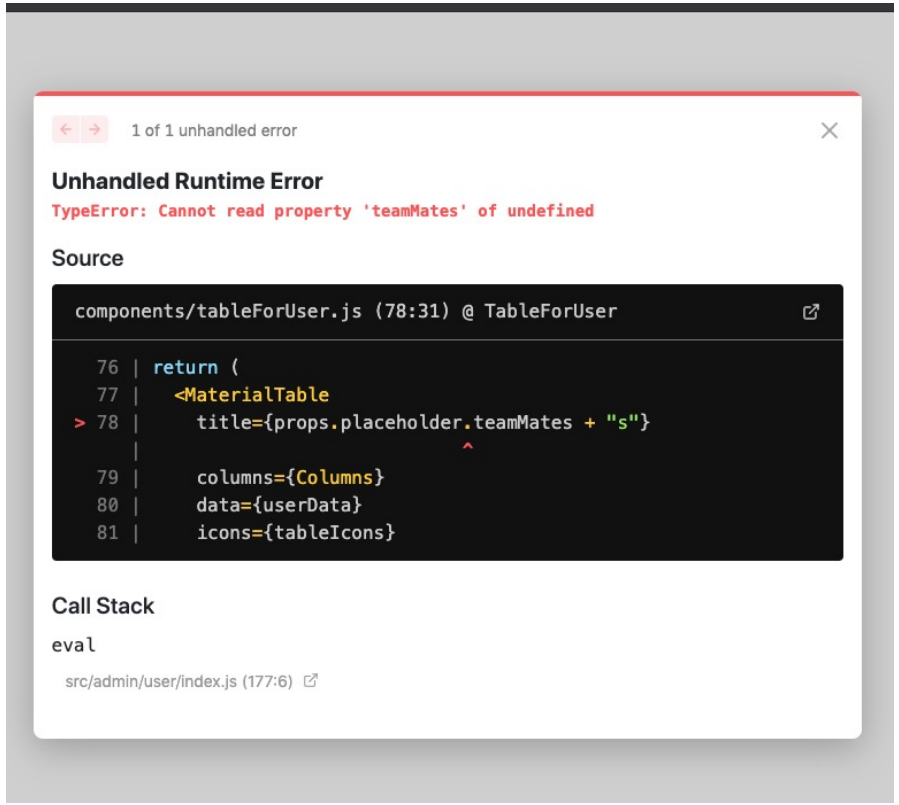
backend	core	26.78.129.90	30 minutes

Save Changes Cancel

Development vs Production

- IP only determines how your application can be accessed
- You can access your local website within the same network
  - Just need to run `npm run dev –H 0.0.0.0`
- Development vs production is about how you run your application!

Development vs Production



https://github.com/vercel/next.js/discussions/12956

Development vs Production

- Uses a test database
  - Sometimes even a smaller engine like SQLite
- Upon errors, the actual stack trace is shown
- Run with a development server

- Uses a real database with user data: Postgres, MySQL, …
- Just a generic 500 or 404 is shown
- Run with a real webserver

Deploying a Next.js project

General notes

- Important: make sure to install the latest version of node
  - The one immediately available on apt is not necessarily up-to-date

- Clone your code on your server
  Install dependencies and deploy migrations
  ```
  npm install
  npx prisma generate
  npx prisma migrate deploy
  ```

- Your app can start via `npm run dev` and accessed via the server's IP address
  - But it's still the development mode

Production env

- Some of developments env variables need to be overridden!

- Examples

  `DATABASE_URL`
  `ACCESS_TOKEN_SECRET`
  `REFRESH_TOKEN_SECRET`
  `GOOGLE_MAPS_API_KEY`

Production settings

- create a .env file and write all settings to that file
  - Create separate files like .env.production and .env.development for overrides

- Frontend env variables should start with `NEXT_PUBLIC_`
  - They will be shared with browser.

- Note: The env files must not be on git!
  - Do push the list of keys (e.g., env.sample) to git!

Development mode

- So far, we've run Next.js apps with `npm run dev`
  - This is the development mode
- Hot reload
  - No need to restart the server when a file changes
- Error tracing and overlay
- TS and JSX files compiled to JS on the fly

Production build

- You should build your app for production
  - Run `npm run build`
- Compiles and bundles all files
- Minifies JavaScript, and performs several optimizations
  - Pre-generates and caches everything it can
- As a result, the app is much faster in production

Production build

- Production build is generated in the .next directory
  - Only pure JS files: no TypeScript, no JSX
- Contains server/, static/, cache/, etc.
- Run the production application via `npm start`

Next.js production Dockerfile (basic)

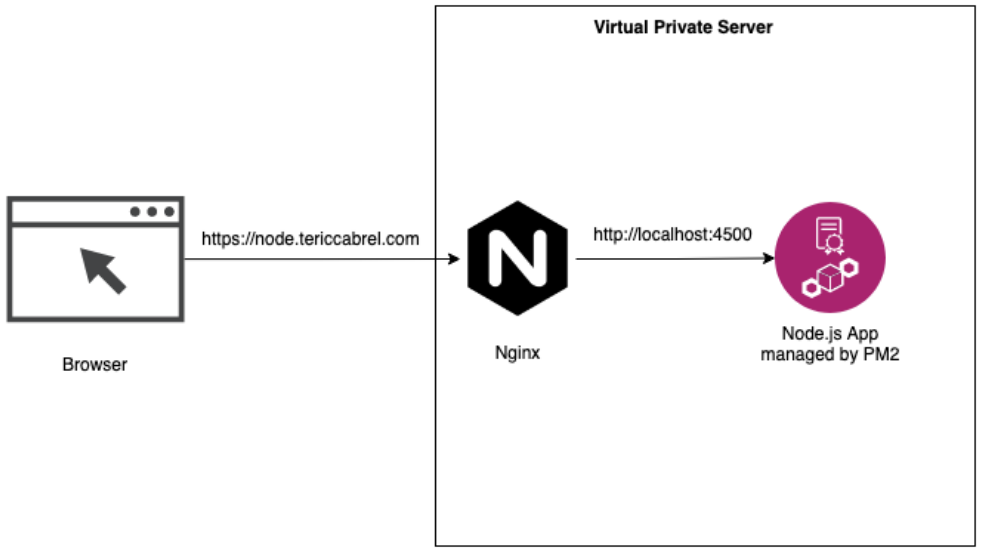
```
FROM node:20-alpine

WORKDIR /app

COPY . .

RUN npm install


RUN npx prisma generate

RUN npm run build

EXPOSE 3000
ENV NODE_ENV=production

CMD npm start
```

Note: Use .dockerignore to exclude unnecessary files (e.g., node_modules)

Process Manager 2

- In production, we do not tend to directly run the Node server

- Instead, the codes run via Process Manager 2 (PM2)



Source: https://blog.tericcabrel.com/deploy-a-node-js-application-with-pm2-and-nginx/

PM2 advantages

- Managing lifecycle:
  - Makes sure the app is running continuously
  - Restart the application after crash
  - Can even start the app on reboot

- Scaling the application
  - Configure the number of workers and cores

- Monitoring
  - Access to logs and metrics

Installing PM2

- Install via

```
sudo npm install -g pm2
```

- Start the app via

```
pm2 start "npm start" --name nextapp
```

- Daemonize the application via

```
pm2 startup
pm2 save
```

PM2 commands

- Check the status of apps
  ```
  pm2 list
  ```
- View logs
  ```
  pm2 logs nextjs-app
  ```
- Monitoring
  ```
  pm2 monit
  ```
- Restart the app
  ```
  pm2 restart nextjs-app
  ```

Next.js production Dockerfile (optimized)

```
FROM node:20-alpine AS builder

WORKDIR /app

RUN npm install -g pm2

COPY package*.json ./
RUN npm install

COPY . .
RUN npx prisma generate

RUN npm run build

# Only copy the necessary files to the production image
FROM node:20-alpine AS runner
WORKDIR /app

# Copy the built Next.js application from the builder stage
COPY --from=builder /app/.next ./.next
COPY --from=builder /app/public ./public
COPY --from=builder /app/package.json ./

# Do not install devDependencies
RUN npm install --only=production

EXPOSE 3000
ENV NODE_ENV=production

CMD pm2-runtime start npm -- start
```

Webserver

- It's time to install a webserver
  - Examples: Nginx, Apache
- Listens on ports 80, 443, etc.
- Serves multiple web applications on the same computer
- Has many security features

Deploy with Nginx

- Create your config file at /etc/nginx/sites-available/<project_name>
- Make a symbolic link to that file at /etc/nginx/sites-enabled/
- Restart Nginx!

```
server {
    listen 80;
    server_name csc309.xyz www.csc309.xyz;

    location /_next/static/ {
        alias /home/ubuntu/csc309_proj/.next/static/;
    }

    location /public/ {
        alias /home/ubuntu/csc309_proj/public/;
    }

    location / {
        include proxy_params;
        proxy_pass http://localhost:3000/;
    }
}
```

Static files

- Static files are file/directory access granted to the webserver

- Including static JS, CSS, and HTML files (generated at build), as well as images, fonts, etc.
  - Should be served by the webserver itself!
  - Webserver offers better security and performance

- Could be uploaded to CDNs for even faster retrieval

Additional steps

- You might need to check permissions as well

- It needs to read your static files and iterate your directories
  - Read/execute access to directories of your static files (including parent directories)
  - Read access to all static files

A Next.js Docker compose

- Volumes
  - Database
  - Static files storage

- Services (containers)
  - Database
  - Migration runner
  - Backend code (built and run via pm2)
  - Webserver

Vercel

Visit https://vercel.com/

- The creator of Next.js
- Has an incredibly straightforward deployment system
- Simply create an account, connect your repo, and that's it!
- It will build the application and host it on its own servers
  - Automatic deployment per push to the main branch
  - Provides HTTPS, scaling, and various optimizations

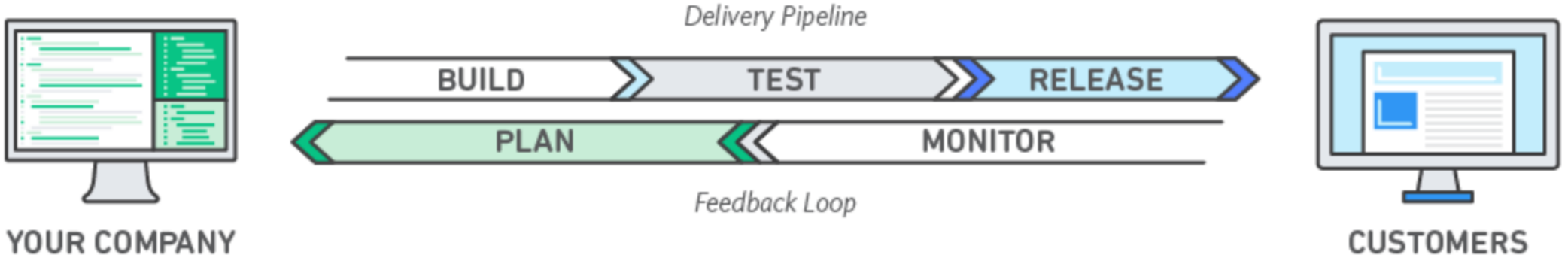
Vercel

- A serverless platform
- Only works for Next.js projects
  - As opposed to the VM-based approach
- Much easier for simple deployments, but much less control over the environment
- For example: cannot host a database with the application
  - Does not come with persistent storage
  - Should use a cloud-provided database

Questions?

DevOps

Visit https://aws.amazon.com/devops/what-is-devops



DevOps

- A set of principles and practices
- Continuous delivery
- Automated build and deployment

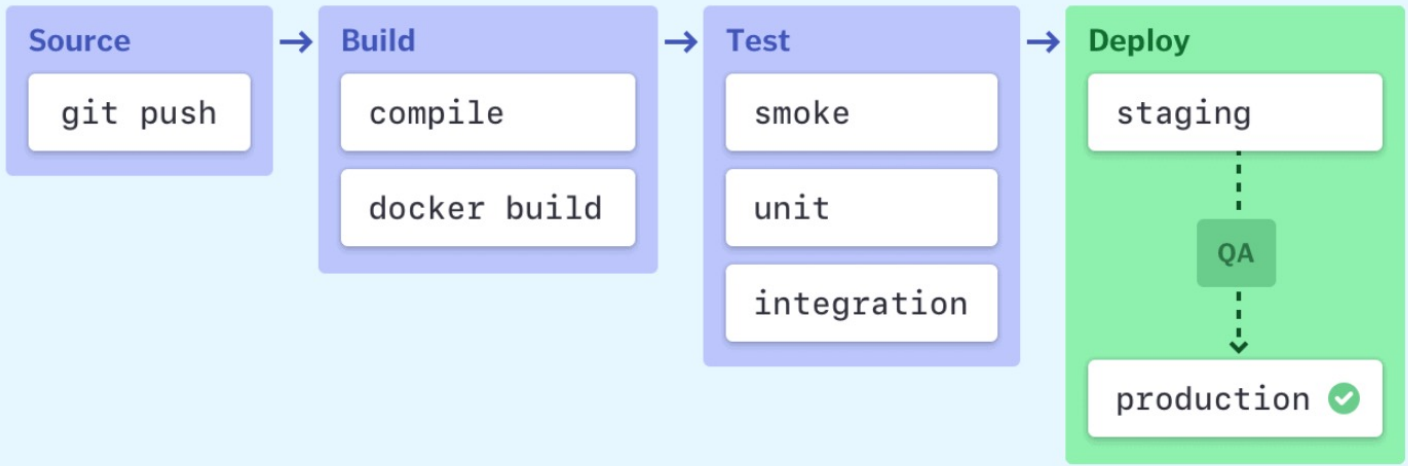
Auto DevOps

- Github and Gitlab support auto DevOps
- Once you push, a pipeline starts
  - Test
  - Build
  - Deploy
- Also called CI/CD (continuous integration & continuous delivery)

Auto DevOps

- Github/Gitlab can run all CI/CD processes through a docker container
- If everything is fine, the docker images will be your application!
- Can be pushed to a registry and pulled in the server

Example gitlab-ci.yaml




```yaml
deploy-staging:
  only:
    - develop
  when: manual
  script:
    - rsync -r ./ $TARGET_DIR
    - source $PY_ENV_PATH
    - cd $TARGET_DIR
    - pip install --upgrade -r requirements/$PIP_FILE.txt
    - python src/manage.py compilemessages
    - python src/manage.py collectstatic --noinput
    - python src/manage.py migrate --noinput
    - python create_env_file.py -o
    - sudo systemctl restart uwsgi
```

Source: https://semaphoreci.com/blog/cicd-pipeline

This course…

- A journey through the world of world wide web!
  - It was a long one, wasn't it?

- It's evolving at the speed of light!
  - Many of these things can go obsolete in 5 years!!
  - New technologies and innovations arrive!

- Hope you've had a fun time :)