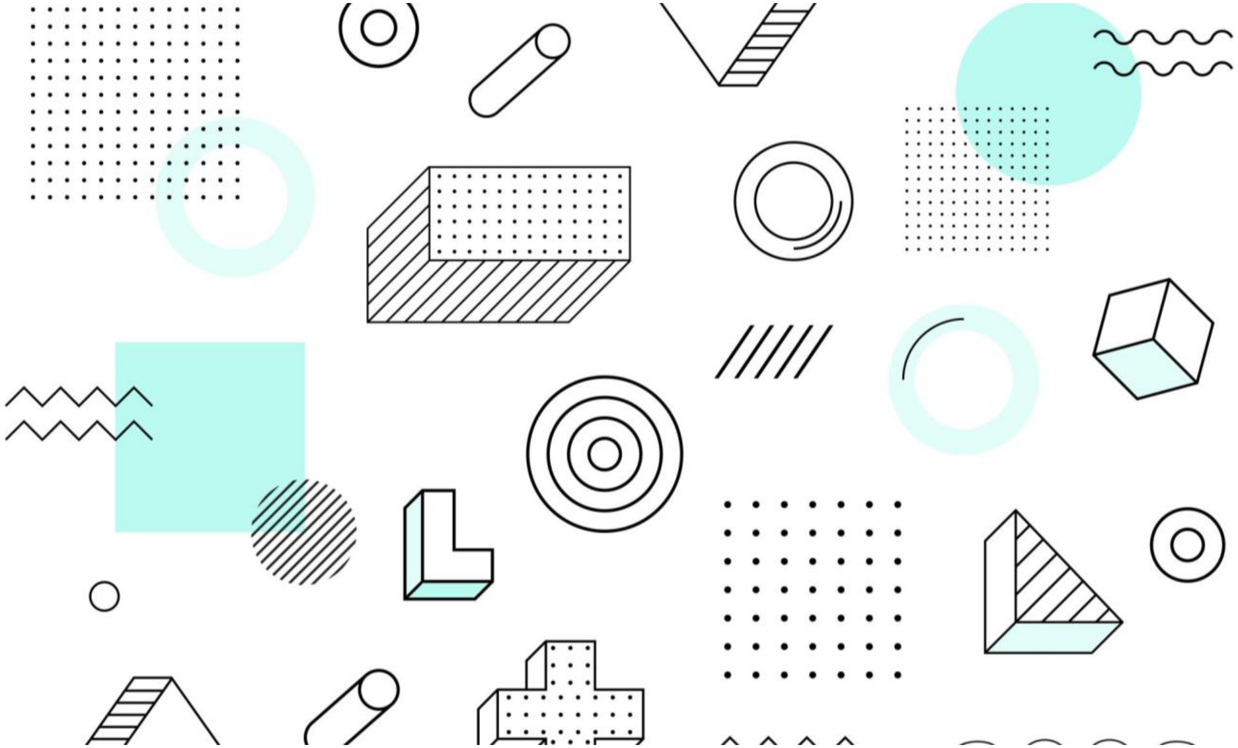
CI/CD and Production featuring: Kubernetes, Azure Devops, and Docker

By Greg Agnew



Docker Prose and Con Artists

Pros

- Dependency Injection
- Extends .NET OS compatibility
- Stable between environments
- Lighter than a Virtual Machine

Cons

- Containerized also means difficult to reach (Volumes and CI/CD)
- Possibly need multiple containers per application
- Might have too many containers in general
- No default port binding
- Version overwriting non-trivial
- Additional overhead for testing

Useful Docker Commands with Docker Hub

- Docker build -t dockerhubuser/dockerhubrepo:tag Dockerfilelocation
- Alt (Same Directory): Docker build --tag dockerhubuser/dockerhubrepo:tag .
- Docker Run --detatch --name nameofcontainer --publish 80:8080 imagename:tag
- Alt imageid for image name, -d, -n, -p
- Docker images
- Docker Push imagename:tag
- Docker ps -a
- Docker-Compose up

Dockerfile (API)

```
FROM mcr.microsoft.com/dotnet/aspnet:5.0 AS base
WORKDIR /app
EXPOSE 80
#EXPOSE 443
#ENV ASPNETCORE_URLS http://*:80

FROM mcr.microsoft.com/dotnet/sdk:5.0 AS build
WORKDIR /src
COPY ["P2Api/P3Api.csproj", "P2Api/"]
COPY ["P2DbContext/P3DbContext.csproj", "P2DbContext/"]
COPY ["P2Main/BusinessLayer.csproj", "P2Main/"]
RUN dotnet restore "P2Api/P3Api.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "P2Api/P3Api.csproj" -c Release -o /app/build

FROM build AS publish
RUN dotnet publish "P2Api/P3Api.csproj" -c Release -o /app/publish

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "P3Api.dll"]
```

Minimum (VM) Image

Default

Dependencies

Files

Test file et. all

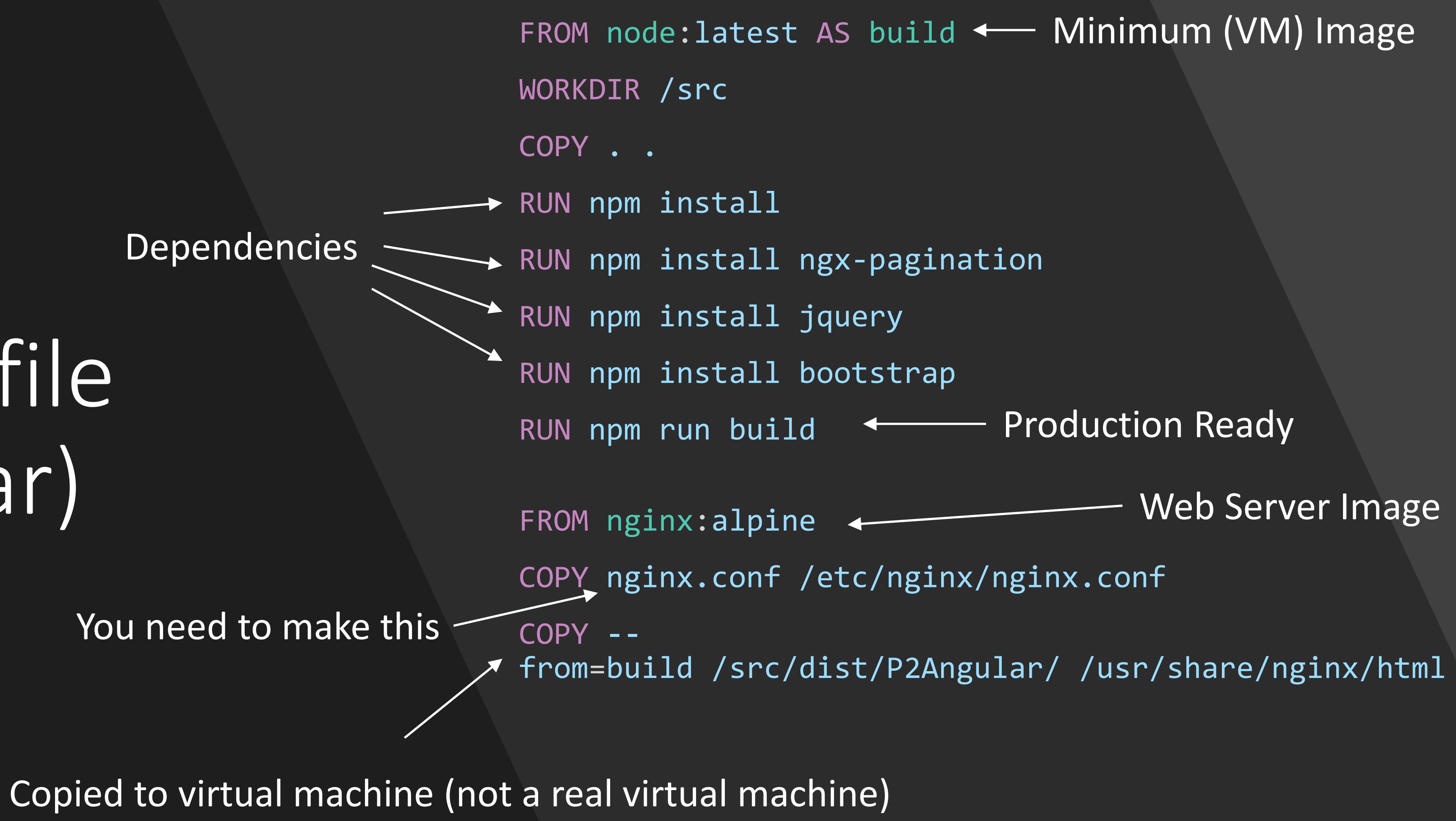
Time Saver

Production Ready

Copied into Virtual Machine

Where to Start Program From

Dockerfile (Angular)



```
FROM node:latest AS build
WORKDIR /src
COPY . .
RUN npm install
RUN npm install ngx-pagination
RUN npm install jquery
RUN npm install bootstrap
RUN npm run build

FROM nginx:alpine
COPY nginx.conf /etc/nginx/nginx.conf
COPY --from=build /src/dist/P2Angular/ /usr/share/nginx/html
```

Minimum (VM) Image

Dependencies

Production Ready

Web Server Image

You need to make this

Copied to virtual machine (not a real virtual machine)

nginx.conf

```
events{}

http {

    include /etc/nginx/mime.types;

    server {
        listen 80;
        server_name localhost;
        root /usr/share/nginx/html;
        index index.html;

        location / {
            try_files $uri $uri/ /index.html;
        }
    }
}
```

Need one per port

Port
(would say 'listen https' for secure)

Aside: Angular Build

```
// This file can be replaced during build by using the `fileReplacements` array.
// `ng build` replaces `environment.ts` with `environment.prod.ts`.
// The list of file replacements can be found in `angular.json`.
```

```
export const environment = {
  production: false,
  urlstat: "https://localhost:44303/api/",
  urlmain: "https://localhost:44307/api/P3/",
  urlgame: "https://localhost:44301/api/Games/",
  urlmainlocalonly: "https://localhost:44307/api/P3/",
};
```

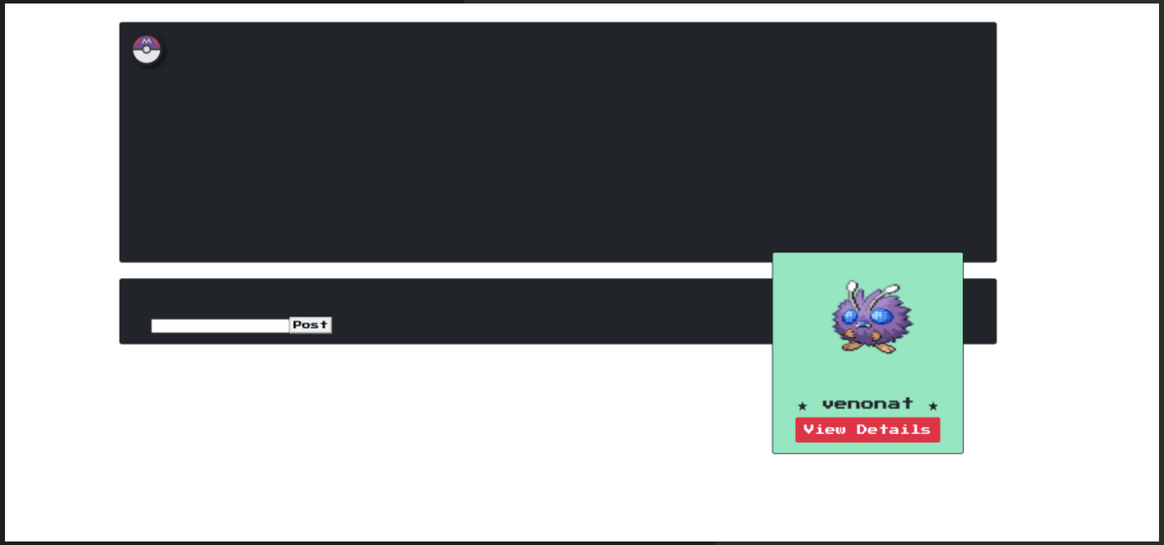
```
export const environment = {
  production: true,
  urlstat: "https://p3pokeloot.com/api/",
  urlmain: "https://p3pokeloot.com/api/P3/",
  urlmainlocalonly:
"https://p3pokeloot.com/api/P3/",
  urlgame: "https://p3pokeloot.com/api/Games/"
};
```

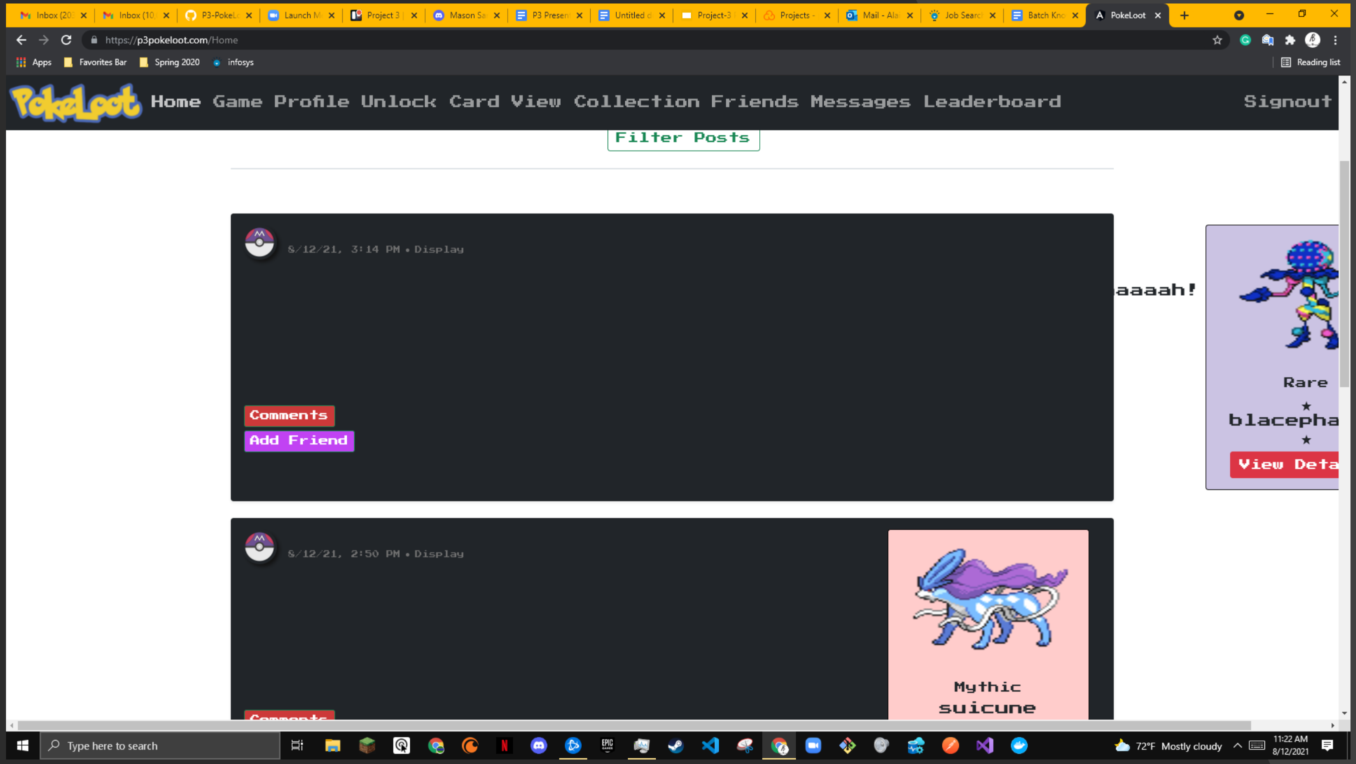
↑

Production

Angular Bootstrap Production Issue

```
<div class="container mt-4 mb-3 shadow-sm p-3 bg-body rounded">
```





Kubernetes Professionals and Constraints

Pros

- Failover
- Rollout
- Ingress

Cons

- Timed Life
- Routing Difficult
- Legacy

Useful Kubernetes Commands

- Kubectl action typeofthing specificthing -n namespace
- Example: kubectl get pods -n pokeloot
- Example: kubectl delete namespace pokeloot
- Example: kubectl describe pods p3/gregious:v12 -n pokeloot
- Example: kubectl apply -f deployment.yml -n pokeloot
- CRUD=>apply,get/describe,apply,delete

~/.kube/config

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: {really long key}
    server: https://p3-pokeloot-dns-02e51f80.hcp.westus3.azmk8s.io:443
  name: p3-pokeloot
- context:
    cluster: p3-pokeloot
    user: clusterUser_06012021Batch_p3-pokeloot
  name: p3-pokeloot
current-context: p3-pokeloot
kind: Config
preferences: {}
users:
- name: clusterUser_06012021Batch_p3-pokeloot
  user:
client-certificate-data: {really long key}
client-key-data: {really long key2}
token: {really long key3}
```

Deployment

apiVersion: Kubernetes version

kind: Deployment, Service, Ingress, ect.

name and label both for pod

selector: important (matching label here)

replicas: for replica set

container name separate from pod name (describe)

image: (defaults to docker hub)

imagePullPolicy: Always looking for new image

containerPort: opens node port and connects them

resources: best practice

Required

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: p3angular
  labels:
    app: p3angular
  annotations:
spec:
  selector:
    matchLabels:
      app: p3angular
  replicas: 2
  template:
    metadata:
      labels:
        app: p3angular
```

(Continued at right)

deployment options

(Lined up with metadata)

```
spec:
  containers:
  - name: p3angular
    image: gregious/p3angular:1278
    imagePullPolicy: Always
    ports:
    - name: http
      containerPort: 8080
    resources:
      requests:
        memory: "64Mi"
        cpu: "50m"
      limits:
        memory: "256Mi"
        cpu: "500m"
```

Kind specific options

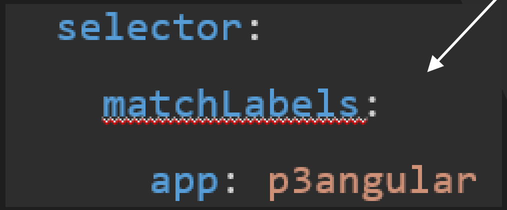
Service

name: Service name
port: port opened on node
targetPort: port on cluser-ip of service
defaults to cluster service if not specified

Separator

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: angular
spec:
  ports:
    - port: 8080
      targetPort: 80
  selector:
    app: p3angular
```

From previous page



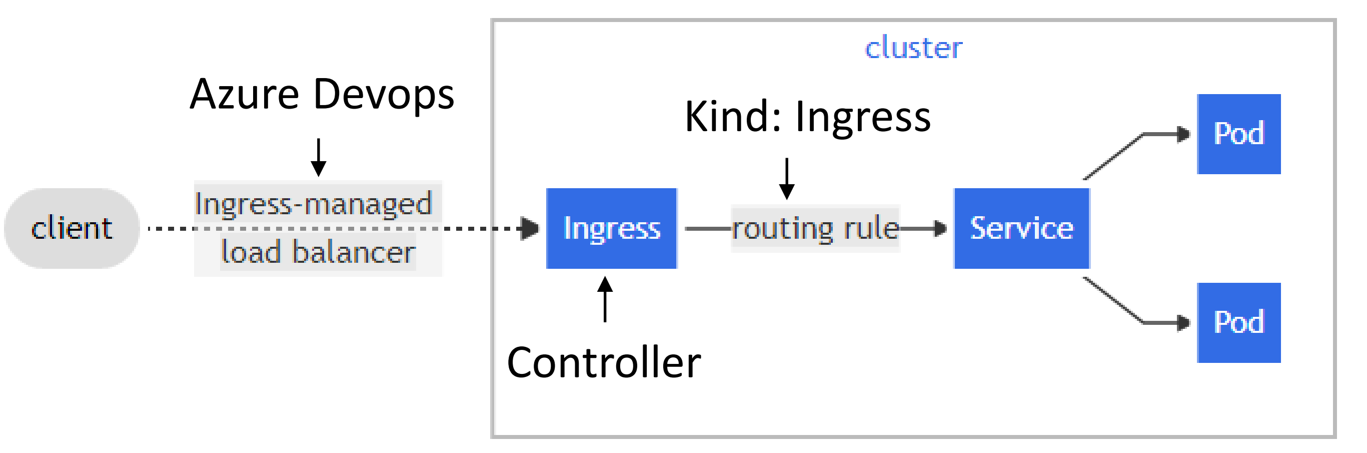
```yaml
selector:
  matchLabels:
    app: p3angular
```

```yaml
apiVersion: v1
kind: Service
metadata:
  name: example-service
spec:
  selector:
    app: example
  ports:
    - port: 8765
      targetPort: 80
  type: LoadBalancer
```

exposed port

Ingress

- Ingress is two things in one.
  - It is a reverse proxy for distributing and routing incoming external traffic requests.
  - It also is a controller that allows Kubernetes to utilize it
- Ingress is better than an external reverse proxy because Kubernetes can directly interface with it.
- Ingress is better than several load balancers because you only need one ip.



Ingress Controller Installation

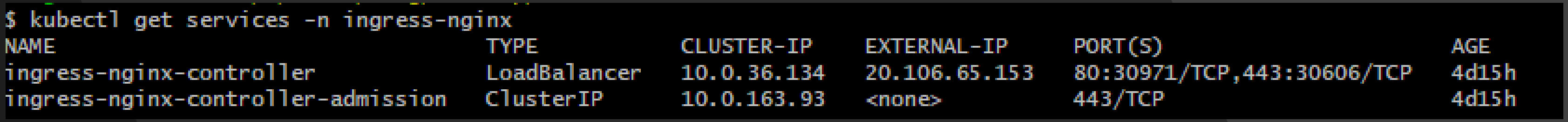
https://kubernetes.github.io/ingress-nginx/deploy/

Azure:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v0.48.1/deploy/static/provider/cloud/deploy.yaml

Defaults to ingress-nginx namespace

Kubectl get services -n ingress-nginx



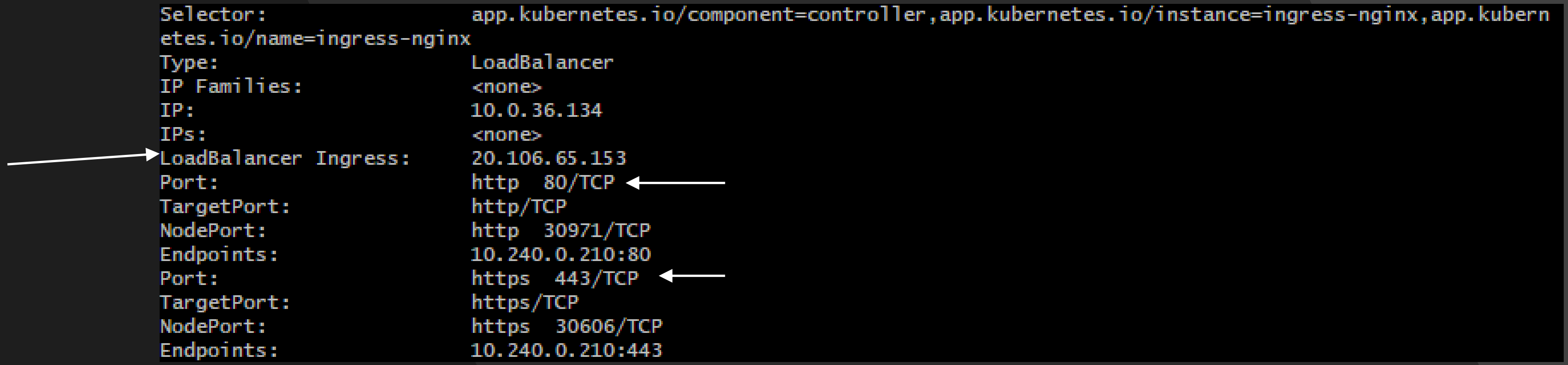
```
$ kubectl get services -n ingress-nginx
NAME                                 TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)                      AGE
ingress-nginx-controller             LoadBalancer   10.0.36.134    20.106.65.153   80:30971/TCP,443:30606/TCP   4d15h
ingress-nginx-controller-admission   ClusterIP      10.0.163.93    <none>          443/TCP                      4d15h
```

Check out the external ip on the controller, that’s your public ip now

Kubectl describe services ingress-nginx-controller -n ingress-nginx

Shows listening on port 80 (http) and port 443 (https)

Is a Load Balancer



```
Selector:                 app.kubernetes.io/component=controller,app.kubernetes.io/instance=ingress-nginx,app.kubern
etes.io/name=ingress-nginx
Type:                     LoadBalancer
IP Families:              <none>
IP:                       10.0.36.134
IPs:                      <none>
LoadBalancer Ingress:     20.106.65.153
Port:                     http  80/TCP
TargetPort:               http/TCP
NodePort:                 http  30971/TCP
Endpoints:                10.240.0.210:80
Port:                     https  443/TCP
TargetPort:               https/TCP
NodePort:                 https  30606/TCP
Endpoints:                10.240.0.210:443
```

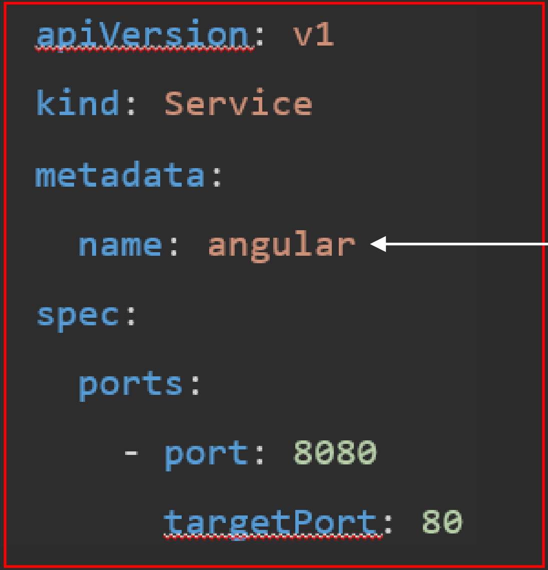
Ingress Resource

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
  namespace: pokeloot
  annotations:
    nginx.ingress.kubernetes.io/enable-cors: "true"
    nginx.ingress.kubernetes.io/cors-allow-methods: "PUT, GET, POST, DELETE, OPTIONS"
    nginx.ingress.kubernetes.io/cors-allow-origin: "20.106.65.153"
    nginx.ingress.kubernetes.io/cors-allow-headers: "*"
    nginx.ingress.kubernetes.io/ssl-redirect: "true"
    nginx.ingress.kubernetes.io/force-ssl-redirect: "true"
    cert-manager.io/issuer: "letsencrypt-prod“
spec:
  tls:
  - hosts:
    - p3pokeloot.com
    secretName: p3pokeloot-prod-certificate
  rules:
  - host: p3pokeloot.com
    http:
      paths:
(continued at right)
```

Public IP
(Could have
used Domain name)

Certificate Issuer

SSL Certificate



```
apiVersion: v1
kind: Service
metadata:
  name: angular
spec:
  ports:
    - port: 8080
      targetPort: 80
```

```
(lined up with paths)
- path: /
  pathType: Prefix
  backend:
    service:
      name: angular
      port:
        number: 80
- path: /api/P3/
  pathType: Prefix
  backend:
    service:
      name: main
      port:
        number: 80
- path: /api/Games
  pathType: Prefix
  backend:
    service:
      name: game
      port:
        number: 80
- path: /api
  pathType: Prefix
  backend:
    service:
      name: statistic
      port:
        number: 80
```

Azure Devops Main Settings

```
trigger:
- main
pool:
  vmimage: Windows-latest
variables:
  buildConfiguration: 'Release'
```

```
trigger:
- main
pool: YourPC

variables:
  buildConfiguration: 'Release'
```

Alt for local

Azure Devops Sonar Cloud and Docker

```
 Sonar Cloud

  - task: SonarCloudPrepare@1

      inputs:

        SonarCloud: 'MainAPI_SonarCloud'

        organization: 'p3pokeloot'

        scannerMode: 'MSBuild'

        projectKey: 'p3pokeloot_Pokeloot_MainAPI'

        projectName: 'p3pokeloot_Pokeloot_MainAPI’

Build something

Where did it go?
```

```
# Main API Docker container
- stage: BuildMain
  displayName: MainDockerBuildandPush
  jobs:
  - job: Build
    displayName: Build
    steps:
    - task: Docker@2
      inputs:
        containerRegistry: 'DockerHub'
        repository: 'gregious/p3mainapi'
        tags: '$(Build.BuildId)'
        command: 'build'
        Dockerfile: '$(Build.SourcesDirectory)/P2Project/P2Main/Dockerfile'
    - task: Docker@2
      inputs:
        containerRegistry: 'DockerHub'
        repository: 'gregious/p3mainapi'
        tags: '$(Build.BuildId)'
        command: 'push'
        addPipelineData: false
        addBaseImageData: false
```

tag

Deploy to docker hub

Azure Devops Building API (From P2)

```
stages:
# Main API Build, Test, Publish to Sonar Cloud
- stage: APIMainSonarCloud
  jobs:
  - job: apijob
    steps:

    - task: SonarCloudPrepare@1
      inputs:
        SonarCloud: 'MainAPI_SonarCloud'
        organization: 'p3pokeloot'
        scannerMode: 'MSBuild'
        projectKey: 'p3pokeloot_Pokeloot_MainAPI'
        projectName: 'p3pokeloot_Pokeloot_MainAPI'
    - task: DotNetCoreCLI@2
      displayName: buildproj
      inputs:
        command: 'build'
        projects: '$(Build.SourcesDirectory)/P2Project/P2Main/P2Main.sln'
        arguments: '--configuration $(buildConfiguration)'
```

Sonar Cloud

Build

Test

(Continued from left and ‘- task:’ aligned with ‘- task:’)

```
- task: DotNetCoreCLI@2
  displayName: testproj
  inputs:
    command: 'test'
    projects: '$(Build.SourcesDirectory)/P2Project/P2Main/P2Main.sln'
    arguments: '--configuration $(buildConfiguration) --collect "Code Coverage"'
    #workingDirectory: '$(Build.SourcesDirectory)/P2Project/P2Main'

 - task: PublishCodeCoverageResults@1
  inputs:
    codeCoverageTool: 'Cobertura'
    summaryFileLocation: '**/coburtura/coverage.xml'

 - task: SonarCloudAnalyze@1
 - task: SonarCloudPublish@1
```

Irrelevant for Sonar Cloud

Publish is optional but enables some Sonar Cloud stuff

Azure Devops Building Angular (From P2)

```
#Build Angular for Sonar Cloud
- stage: Angular
  jobs:
  - job: angularjob
    steps:
    - task: NodeTool@0
      inputs:
        versionSpec: '14.x'
      displayName: 'Install Node.js'

    - task: Npm@1
      inputs:
        command: 'custom'
        workingDir: '$(Build.Repository.LocalPath)\P2Project\P2Angular'
        customCommand: 'install -g @angular/cli'

    - task: Npm@1
      inputs:
        command: 'install'
        workingDir: '$(Build.Repository.LocalPath)\P2Project\P2Angular'
```

Install Node

Install Angular

Install NPM

(Continued from left)

Sonar Cloud

Test location from Angular

```
- task: SonarCloudPrepare@1
  inputs:
    SonarCloud: 'Angular_SonarCloud'
    organization: 'p3pokeloot'
    scannerMode: 'CLI'
    configMode: 'manual'
    cliProjectKey: 'p3pokeloot_Pokeloot_Angular'
    cliProjectName: 'p3pokeloot_Pokeloot_Angular'
    cliSources: '$(Build.Repository.LocalPath)\P2Project\P2Angular\src'
    extraProperties:
'sonar.javascript.lcov.reportPaths=$(Build.Repository.LocalPath)\P2Project\P2Angular\coverage\P2Angular\lcov.info'

  - task: CmdLine@2
    inputs:
      script: 'ng build'
      workingDirectory: '$(Build.Repository.LocalPath)\P2Project\P2Angular'

  - task: Npm@1
    displayName: 'NPM Test'
    inputs:
      command: 'custom'
      workingDir: $(Build.Repository.LocalPath)\P2Project\P2Angular
      customCommand: 'run test-headless'

  - task: SonarCloudAnalyze@1

  - task: SonarCloudPublish@1
```

Production build

Test

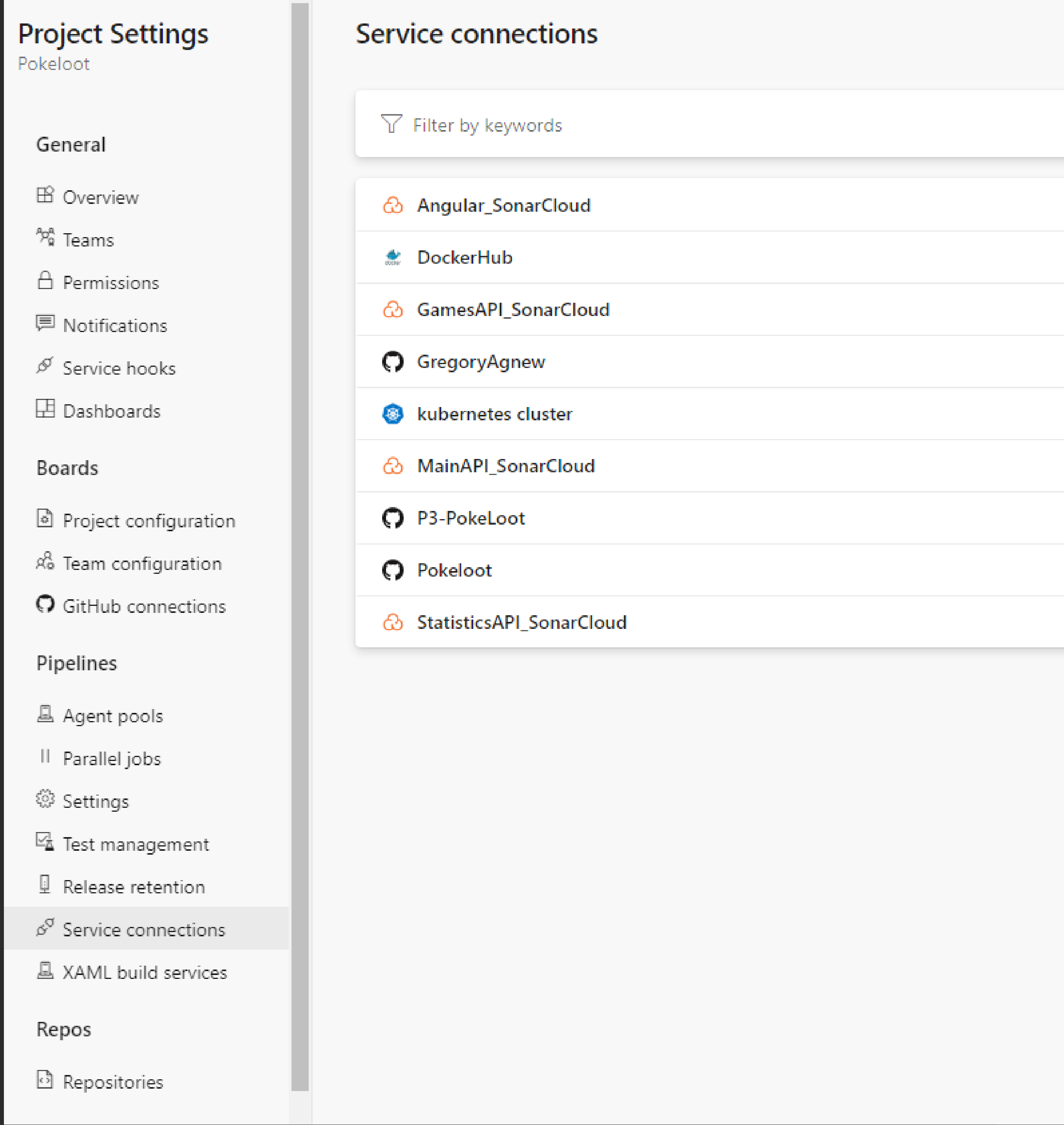
Azure Devops Kubernetes Manifest

```
# Deploy images to kubernetes
- stage: DeployKube
 displayName: Deploy to Kubernetes
 jobs:
 - job: Deploy
  displayName: Deploy
  steps:
  - task: KubernetesManifest@0
   inputs:
    action: 'deploy'
    kubernetesServiceConnection: 'kubernetes cluster'
    namespace: 'pokeloot'
    manifests: '$(Build.SourcesDirectory)/deployment.yml'
    containers: |
     'gregious/p3mainapi:$(Build.BuildId)'
     'gregious/p3gamesapi:$(Build.BuildId)'
     'gregious/p3statisticsapi:$(Build.BuildId)'
     'gregious/p3angular:$(Build.BuildId)'
```

Location of manifest

Azure Devops Supplementals

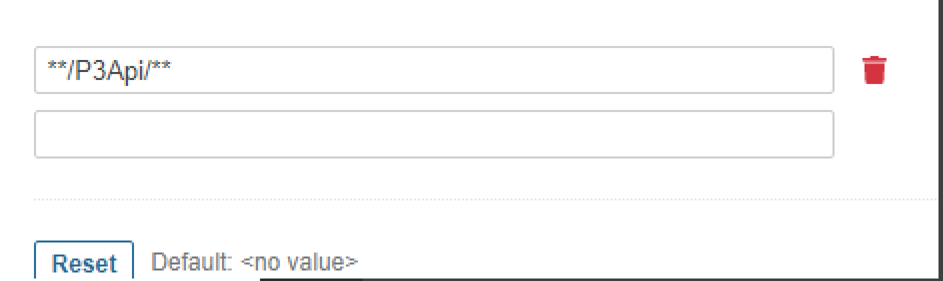
Need various connections to github, sonar cloud, Kubernetes, etc.



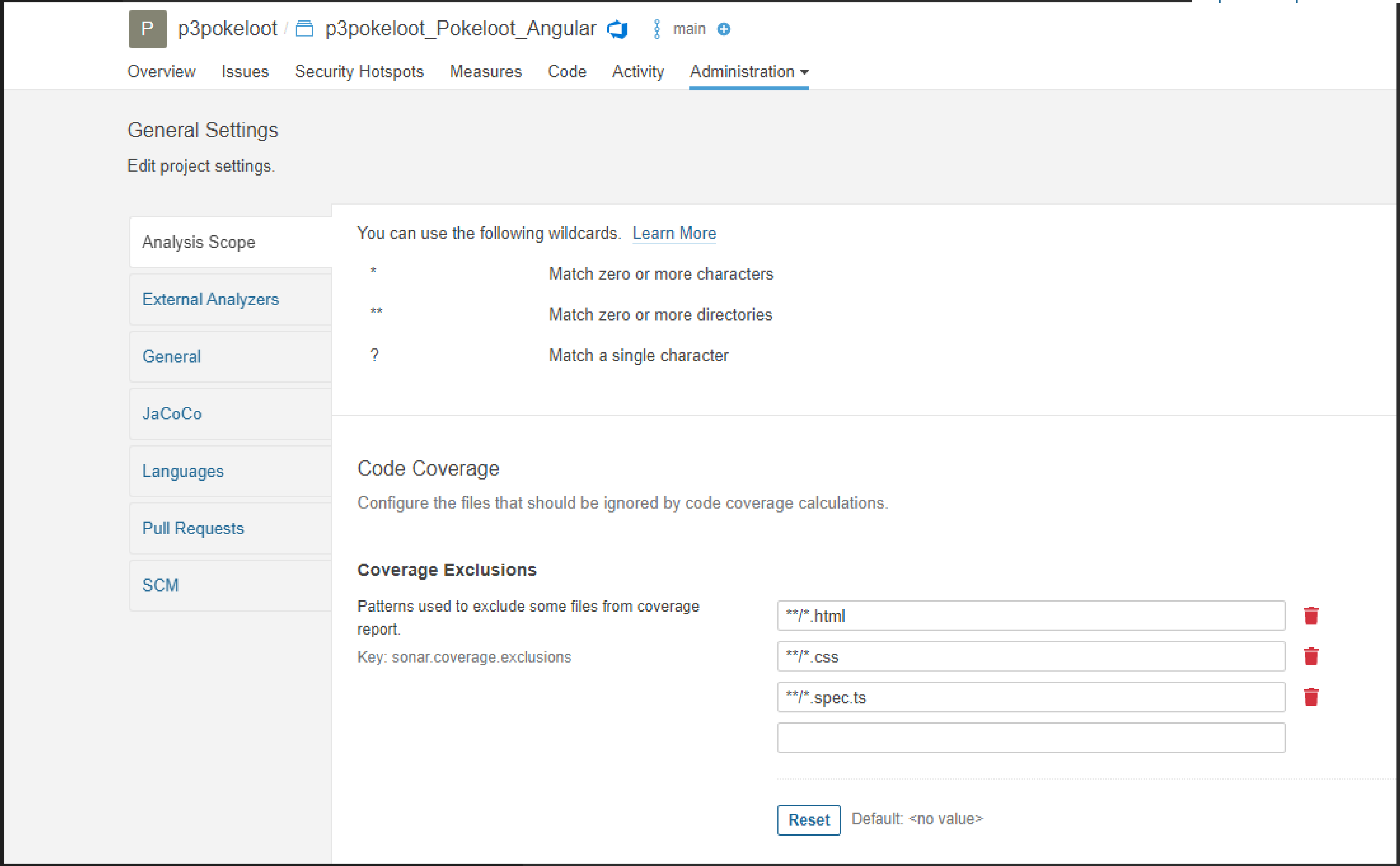
Service connections

- Angular_SonarCloud
- DockerHub
- GamesAPI_SonarCloud
- GregoryAgnew
- kubernetes cluster
- MainAPI_SonarCloud
- P3-PokeLoot
- Pokeloot
- StatisticsAPI_SonarCloud

Sonar Cloud Code Exclusions



/P3Api/



General Settings

Edit project settings.

You can use the following wildcards. Learn More

*	Match zero or more characters
**	Match zero or more directories
?	Match a single character

Code Coverage

Configure the files that should be ignored by code coverage calculations.

Coverage Exclusions

Patterns used to exclude some files from coverage report.

Key: sonar.coverage.exclusions

**/*.html

**/*.css

**/*.spec.ts

El Fin

Did I miss anything?