Contents

[Site Maps 1](#_Toc36495100)

[CI/CD Setup Summary 3](#_Toc36495101)

[Infrastructure - Docker Registry Overview 4](#_Toc36495102)

[Infrastructure – CI Server Overview 4](#_Toc36495103)

[Docker Registry – Push to Registry and tag with build number: 4](#_Toc36495104)

[Docker Daemon - Register Insecure Registry**:** 4](#_Toc36495105)

[Docker Configure SSL by configuring unsigned cert within a Java Container instance: 5](#_Toc36495106)

[Docker FAQ / Notes 6](#_Toc36495107)

[Example CI Script: Docker 6](#_Toc36495108)

[Copying files in and out of running Docker instance 6](#_Toc36495109)

[Docker CLI Notes 6](#_Toc36495110)

[Docker – Show Images 6](#_Toc36495111)

[Docker – Show Running Instances 6](#_Toc36495112)

[Docker - Saving Disk Space 6](#_Toc36495113)

[TeamCity FAQ / Notes 7](#_Toc36495114)

[Dotnet FAQ / Notes 7](#_Toc36495115)

RED = TO DO

Site Maps: edit (needs admin):"C:\Windows\System32\drivers\etc\hosts" OR Add DNS Entry

* Build Registry: (Host entry: *127.0.0.1 dev-registry)*
  + <http://localhost:55000/v2/_catalog>
  + <http://dev-registry:55000/v2/_catalog>
  + <http://localhost:55000/v2/gen/tags/list>
* CI Server:
  + [http://localhost: 8111](http://localhost:55000/v2/_catalog)

CI/CD Setup Summary**:** See Pipeline Design.pptx

* Setup:
  + *Infrastructure - Docker Registry Overview*– infra\registry
  + *Infrastructure – CI Server Overview* (pick one)
    - GitLab – infra\gitlab
    - TeamCity – infra\teamcity – note: compose file includes docker integration and CLI
    - Jenkins - infra\jenkins
* Per Project (Road Map to Pipeline):
  + Establish CM
    - Sonalysts BitBucket (IT provides Account, Space and Master Repo created by DevOps)
    - Perforce (DevOps provides user Account, Space and Master Repo created by DevOps)
  + Traditional App: Write IaaC, Build, Test
    - No Containers using Traditional Process = C#/Publish -> Runtime
    - Create build artifacts, for example: MSBUILD (.sln and csproj) or Maven (Java and packages)
  + Add Docker file and CI Scripts to Traditional App
    - Leverage Containers as Dumb Build Agents
    - *Example CI Script: Docker*
    - Runtime Container = Prepublished running in container
    - SDK container = dependencies combined in container = Dumb Build Agents
    - Multi-staged and Docker Compose w/ dependences within containers
    - Generate an Docker Image
  + *Docker Registry – Push to Registry and tag with build number*:
  + Run Integration tests
  + Deploying to Testing environments or Production
* Performance Testing

## Infrastructure - Docker Registry Overview

* Configure using valid certificate: <https://docs.docker.com/engine/security/certificates/>
* Setup and Deploy - Run Docker Registry in detached mode:
  + aspnetcore-generator-api\infra\registry
  + docker-compose up -d
* Add Alias for registry to DNS or HOSTs file: *dev-registry*
* Docker Daemon - Register Insecure Registry:
* Docker Registry – Push to Registry and tag with build number:

## Infrastructure – CI Server Overview

* Pick a CI Server (infra\gitlab, infra\teamcity or infra\jenkins)
* Setup and Deploy - Run rocker file in detached mode:
  + See \infra\teamcity
  + docker-compose up -d
* Enable SSL – see *Docker Configure SSL by configuring unsigned cert within a Java Container instance*

## 

## Build Images:

* + *cd \aspnetcore-generator-api*
  + *docker build -t testing .*
  + *docker run –rm –it –p 8080:80 testing*

## Docker Registry – Push to Registry and tag with build number:

* + docker build -t "localhost:55000/gen:man\_ci-01" --no-cache .
  + docker tag *aspnetcore/generator:multi* dev-registry:55000/aspnetcore/generator:muli

Tag ^^ existing^^ ^^ local repo alias:tag^^

* + See new tag: Docker image ls
  + docker push dev-registry:55000/aspnetcore/generator:muli  
    Pushes to dev-registry:55000

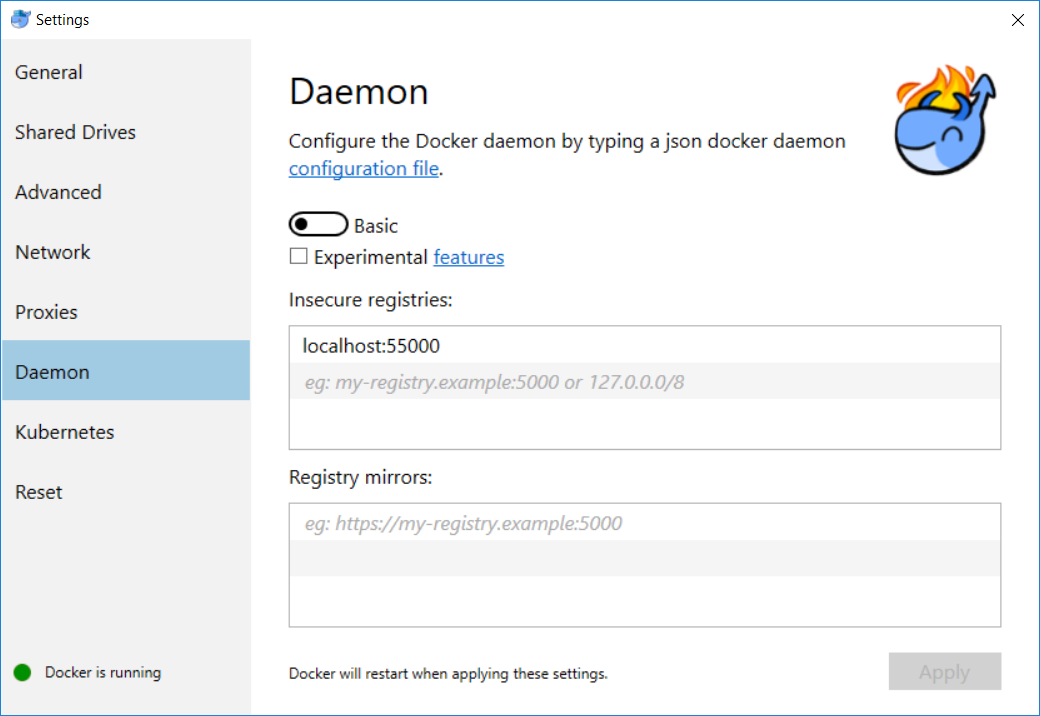
NOTE: Ensure registry is started… <http://localhost:55000/v2/_catalog>

* + See in registry browse to: <http://localhost:55000/v2/gen/tags/list>

## Docker Daemon - Register Insecure Registry**:**

Note: This is only necessary when Registry is not secure i.e. uses self-signed cert

* Temporary configure Daemon it’s OK to connect without TLS/HTTPs:
  + Open your docker settings (right click menu tray whale) and go to Daemon tab. Add
    - dev-registry:55000



* You will be asked to Share access to create a volume: enter SONALYSTS]kaubin with domain login
* To ensure working:
  + docker info
  + look for “Insecure Registries: dev-registry:55000 127.0.0.0/8”

## 

## Docker Configure SSL by configuring unsigned cert within a Java Container instance:

Note: this is the procedure for enabling SSL within TeamCity Docker Instance

* Get the self-signed certificate from the website: <https://bitbucket.sonalysts.com/>
* Put it into some (e.g. ~/git-certs/cert.cer) file – export as Base-64 encoded x.509 (.CER)
* Open Command Prompt
* Use following command to determine the docker container tag
  + C:\>*docker ps*
* Copy cert.cer file to Docker Instance:
  + C:\>*docker cp "C:\Users\kaubin\Desktop\SonoBitBucketCA.cer" 30da31561ba4:/git-certs/SonoBitBucketCA.cer* ^^cert file path *docker\_instance*:*path\_within\_running\_instance*
* Connect to Docker Instance: <https://phase2.github.io/devtools/common-tasks/ssh-into-a-container/>
  + C:\>docker exec -it 30da31561ba4 bin/bash
* Import cert so Java is happy (see <https://confluence.jetbrains.com/display/TCD9/Using+HTTPS+to+access+TeamCity+server#UsingHTTPStoaccessTeamCityserver-ConfiguringJVMforauthenticationwithservercertificate> Configuring JVM section):
  + root@30da31561ba4:/# keytool -importcert -file ./git-certs/cert.cer -keystore ./opt/java/openjdk/jre/lib/security/cacerts
  + Enter keystore password: changeit
* NOTE: By default, Java keystore is protected by password: "changeit"

# Registry FAQ / Notes:

Registry Web Paths:

http://www.my-registry:55000/v2/\_catalog

http://www.my-registry:55000/v2/\_catalog

Docker FAQ / Notes**:**

## Example CI Script: Docker

* Docker File: \api\Dockerfile
* Within CI Server Web Interface:
  + add Build Step: Command Line:

image="localhost:55000/gen:ci-%build.number%"

docker build -t $image .

docker push $image

docker image rm $image

docker build -t "dev-registry:55000/man:l-build" .

Copying files in and out of running Docker instance: <https://til.codes/copy-file-from-host-machine-to-docker-container/>

* docker cp foo.txt mycontainer:/foo.txt
* docker cp mycontainer:/foo.txt foo.txt

Docker CLI Notes**:**

Docker – Show Images

docker image ls list images within repositories

Docker – Show Running Instances

docker ps running processes

Docker - Saving Disk Space:

docker image prune

docker rm IMAGE\_SHA

Creating Docker build scripts:

Create Docker file, running it with a tag (“Testing” below), list files to inform/customize dockerignore:

# RUN ls -alR

# > docker build .

# OR

# Note: ls -alR requires Git Bash (i.e. MINGW) or Linux subsystem for windows

# > docker build -t testing .

# > docker run --rm testing ls -alR

Running Docker file from local machine: docker run --rm -it -p 8080:80 testing

OR

Running Docker file from remote machine: docker run --rm –it –p 8080:80 my-registry:55000/get:ci-9

*Note: be sure remote machine has registered unsecure registries with Docker Deamon. Additionally, my-registry need be mapped within the DNS.*

Ensure proper cleanup after exit of docker-compose image and that Image is rebuilt

Docker-compose up --force-recreate --abort-on-container-exit –build

Docker-compose down

List files within an image

docker run –rm –it –entrypoint=bash localhost:55000/gen:integration-10

root@hast:/integration# ls -al

TeamCity FAQ / Notes**:**

ERROR: Test connection failed in Secure\_Coding\_GSSP / Dot\_NET  
List remote refs failed: javax.net.ssl.SSLHandshakeException: sun.security.validator.ValidatorException: PKIX path building failed: sun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target

SOLUTION: Two Options to deal with this issue:

* Workaround: Disable Git SSL. Works in a pinch but not the solution.

*git* config --global http.sslVerify false

* Best Solution:
  + NOT NESSESARY… kept for future reference  
    Set git to trust this certificate using http.sslCAInfo parameter
    - git config --system http.sslCAPath /git-certs/SonoBitBucketCA.pem
    - git config --system credential.manager manager
    - git config --system credential.modalprompt true

SOURCES:  
<https://stackoverflow.com/questions/9072376/configure-git-to-accept-a-particular-self-signed-server-certificate-for-a-partic>

<https://confluence.atlassian.com/fishkb/unable-to-clone-git-repository-due-to-self-signed-certificate-376838977.html>

<https://confluence.jetbrains.com/display/TCD9/Using+HTTPS+to+access+TeamCity+server#UsingHTTPStoaccessTeamCityserver-ConfiguringJVMforauthenticationwithservercertificate>

Jenkins FAQ / Notes**:**

docker pull jenkins/jenkins:lts

port forward slave port image name image source

CLI Appraoch: docker run –p 2119:8080 –p 50000:50000 –name jenkins-master jenkis/jenkisn/lst

OR

Docker-compose with volume:

*version: "3.7"*

*services:*

*jenkins:*

*#restart: unless-stopped #always none on-failure unless-stopped*

*image: jenkins/jenkins:lts*

*container\_name: jenkins-master*

*user: jenkins*

*volumes:*

*- ./data-jenkins:/var/jenkins\_home*

*- /var/run/docker.sock:/var/run/docker.sock*

*environment:*

*JENKINS\_HOST\_HOME: "/data/jenkins"*

*ports:*

*- "2119:8080"*

*- "5000:5000"*

*- "50000:50000"*

NOTE: There will be an admin password for intial setup. Its within the shell readout when running the container. If you miss it. The passwords found within the container file structure at /var/jenkins\_home/secrets/initialAdminPassword. To access the container structure:

// to unlock initial installation, specify random password provided within container  
docker exec –it Jenkins-master /bin/bash // exec into the container

cd /secrets

cat initalAdminPasswd // look at file contents to find password

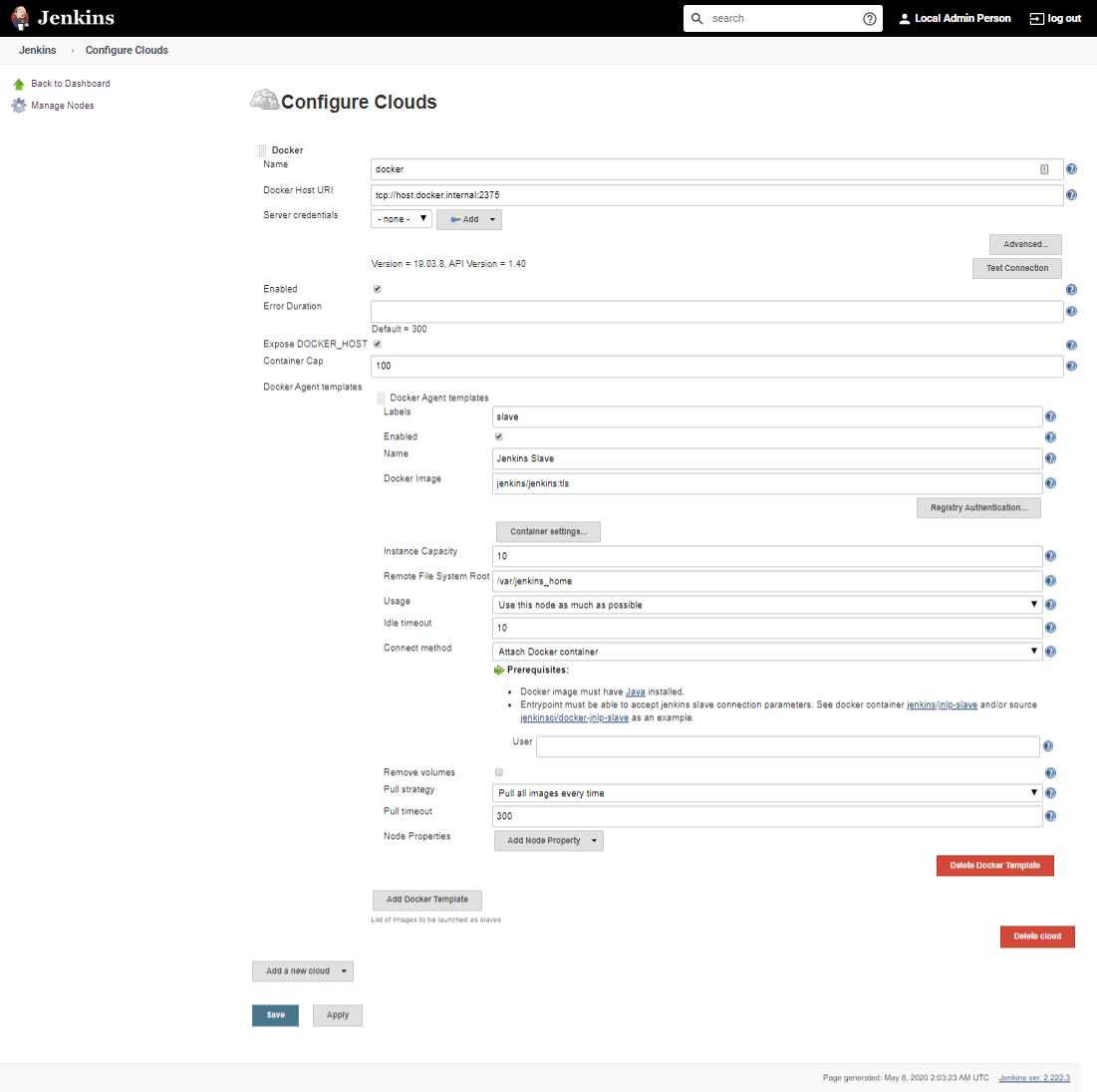
Browse to <http://localhost:2119/>, Enter the password from initialAdminPassword

Plugins: Select custom… leave defaults but add:

* Build Tools:
  + MSBUILD
  + Node.JS
* Build Analysis and Reporting
  + HTML Publisher
  + Junit
  + XUnit
* Source Control Management
  + Git Parameter
  + P4 (Perforce Client)
* User Management and Security:
  + Role-based Authorization Strategy
  + Active Directory
  + Authorize Project
* Notifications and Publishing
  + Email Extension Template

Configure Jenkins Cloud:

* Install Docker Plugin
  + Login as Administrator
  + Click “Manage Plugins”
  + Select “Available” tab
  + Check “Docker”
  + Click “Download now and Install after restart”
* Test Docker API access: <http://localhost:2375/containers/json>
* Apply settings within Image to:
  + Hooks Jenkins to Docker Host (API)
  + Defines agent instances and which image to deploy



Dotnet FAQ / Notes**:**

.NET Traditional App

Creating:

dotnet new xunit –o tests

cd tests

dotnet add reference ../api/api.csproj

Running:

dotnet restore

dotnet test

Docker-compose.yml

* Install Docker
* Create docker compose file docker-compose.yml
  + Determine docker engine versions\: Docker version
  + <https://docs.docker.com/compose/compose-file/> to determine correct version
  + Example:

version: "3.7"

Services:

Web:

Image: nginx

Ports:

-9090:80

Database:

Image: redis

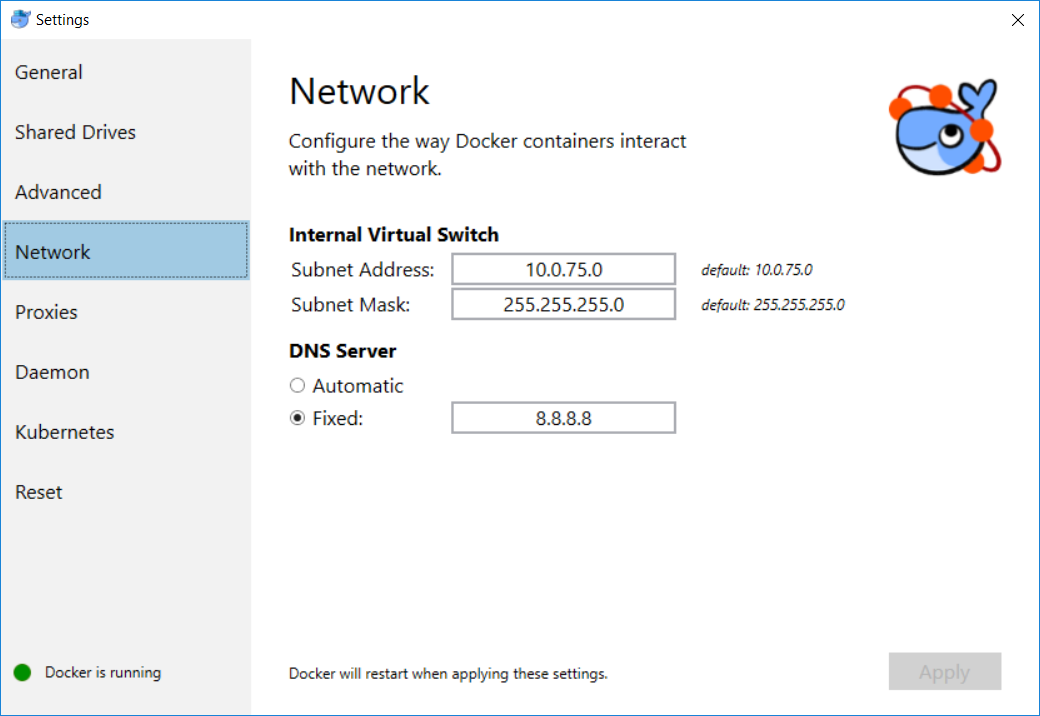
* Check the validity of docker-compose.yml: docker-compose config
* Start compose file with detached option to run in the background: Docker-compose up –d
* Docker-compose down
* See running containers: Docker ps

Scale services:

Docker-compose up –d –scale database=4

Error:  [**Client.Timeout exceeded while awaiting headers**](https://github.com/docker/for-win/issues/1534)**”**

1. No need to modify Default Switch or DockerNAT via Hyper-V
2. In Hyper-V Manager: Click on MobyLinuxVM settings, ensure network adapter set to DockerNAT (should be default).
3. Open your docker settings (right click menu tray whale) and go to network tab. Select Fixed option under DNS Server.(The value will be 8.8.8.8)



GitLab Notes

docker run --detach --hostname gitlab.example.com --publish 443:443 --publish 80:80 --publish 22:22 --name gitlab --restart always --volume /srv/gitlab/config:/etc/gitlab --volume /srv/gitlab/logs:/var/log/gitlab --volume /srv/gitlab/data:/var/opt/gitlab gitlab/gitlab-ce:latest

Kubernetes Notes:  
// Create something new – errors if deployed already use:

kubectl create –f .\api.deployment.yml –save-config

// Create if not exist, otherwise update existing use:

kubectl apply –f ./api.deployment.yml

// Scale Pods Horizontally (imperative)

// Scale the deployment pods to 5

Kubectl scale deployment [deployment-name] --replicas=5

// Scale by referencing the YAML file

kubectl scale -f api.deployment.yml --replicas=5

cd aspnetcore-generator-api\api

kubectl create –f .\api.deployment.yml --save-config

// View the pod status to check that it is ready:

kubectl get pods

//View the Deployment’s status:

kubectl get deployment

// The Deployment automatically manages a ReplicaSet. View the ReplicaSet status:

kubectl get replicaset

// Get existing node status:

kubectl get nodes

// Describe a pods details…

kubectl describe  name-gen-api

// Stop Services

**# Shut down foo.**

**$ kubectl stop replicationcontroller foo**

**# Stop pods and services with label name=myLabel.**

**$ kubectl stop pods,services -l name=myLabel**

**# Shut down the service defined in service.json**

**$ kubectl stop -f service.json**

**# Shut down all resources in the path/to/resources directory**

**$ kubectl stop -f path/to/resources**

// Create API service

kubectl create –f .\api.service.yml --save-config

// Check the Service created (see ports):

kubectl get service name-gen-api

// Configure port forwarding:

*# Note:* use ‘kubectl get pods’ to identify name

*# Change redis-master-765d459796-258hz to the name of the Pod*

kubectl port-forward name-gen-api-d5f885787-8d5h6 8080:80