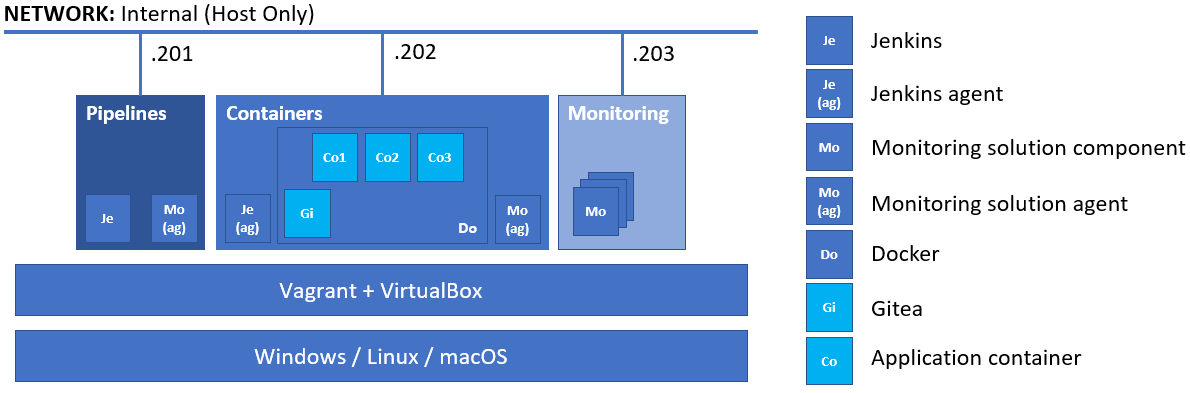
# Exam: DevOps #1 2022.02 (2022.04.03)

### Main goal

You are expected to utilize all or most of the studied products and technologies and create an infrastructure with **three** hosts. Their parameters are up to you to decide *(considering your free resources and the actual distribution of components)*

The goal is to have the whole **infrastructure** as a **file** or **set** of **files**. Then on top of it to create an **automated** **build** **process** which will wait for a hook call by the **source control system** and if there is a change in the project then all related images should be **re-build** and **re-run**

Your solution should look like and follow this structure:



**All** **hosts** should be **provisioned** and **configured** in an automated fashion by utilizing **both** **Vagrant** and **bash** scripts

The **emphasis** should be on **features** usage **demonstration** versus optimal solution

### Rules

Be sure to **follow** the **naming** **conventions** specified in the checklist and in project source files

The tasks execution order should not be derived from the order in which they are listed below. Please note that there are tasks that depend on the successful completion of one or more other tasks

### Proof

You are expected to prepare a short document that outlines the steps you did. It should include all major milestones

If there are any manual steps, you **must** describe them in a free form (including commands if any) in the document. Don’t forget to include some pictures of the important (according to you) steps and of the **result** as well

The document, together with all configuration files and scripts, should be put in a compressed archive and uploaded to [https://softuni.bg/trainings/3670/devops-containerization-ci-cd-monitoring-february-2022#lesson-37521](https://softuni.bg/trainings/3670/devops-containerization-ci-cd-monitoring-february-2022" \l "lesson-37521)

### Tasks

#### Infrastructure (9 pts)

*You are expected to demonstrate knowledge working with* ***Vagrant*** *and* ***VirtualBox***

* (T101, 2 pts) Infrastructure with **three** machines each with a dedicated role
* (T102, 1 pts) All virtual machines named according to the following convention – ***role*.do1.exam**. For example, ***pipelines.do1.exam***
* (T103, 1 pts) All hosts in a dedicated network of your choice. For example, ***192.168.111.0/24***
* (T104, 1 pts) All hosts should have the last octet in their address set according to the picture above
* (T105, 4 pts) At least one host provisioned with the help of **Vagrant** and shell *(inline or external)* script

#### Source Control (8 pts)

*You are expected to demonstrate knowledge working with* ***Gitea****. On the picture it is displayed as* ***Gi****. It is expected that* ***Gitea*** *is run as a container*

* (T201, 2 pts) Installed and working **Gitea**
* (T202, 3 pts) Local project named **exam**, copied from [**https://github.com/shekeriev/fun-facts**](https://github.com/shekeriev/fun-facts)
* (T203, 3 pts) Configured web hook to **Jenkins**

#### Pipelines (19 pts)

*You are expected to demonstrate knowledge working with* ***Jenkins****. On the picture it is displayed as* ***Je***

* (T301, 3 pts) Working base installation of **Jenkins** with configured administrator user
* (T302, 1 pts) Additional (at least one) plugin(s) installed and enabled
* (T303, 1 pts) Added credentials for the **vagrant** user
* (T304, 1 pts) Added credentials for **Docker Hub**
* (T305, 2 pts) Added slave (agent) node (the **Docker** host). *On the picture it is displayed as* ***Je (ag)***
* (T306, 10 pts) А pipeline that has the following stages:
  + (T306.1, 1 pts) gets the project code from **Gitea**
  + (T306.2, 2 pts) builds the images
  + (T306.3, 2 pts) runs the application in test mode *(the front-end component listens on 8080)*
  + (T306.4, 2 pts) tests if the front-end is reachable
  + (T306.5, 1 pts) publishes the images to **Docker Hub**
  + (T306.6, 2 pts) deploys the application in production mode *(port 80)* out of the published images
* (T307, 1 pts) The pipeline should be triggered via a webhook

*Once done, you should commit a few changes (change the contents of the* ***client/code/app.dat*** *file) to prove the whole pipeline + source control setup is working*

#### Monitoring (16 pts)

*You are expected to demonstrate knowledge working with either* ***Prometheus*** *+* ***Grafana*** *or* ***Elastic Stack*** *(****Elasticsearch****,* ***Logstash****, and* ***Kibana****) On the picture it is displayed as* ***Mo***

* (T401, 5 pts) Working base installation of all components of the chosen monitoring solution
* (T402, 3 pts) Deployed monitoring agent *(extractor or beat)* on the **Jenkins** and **Docker** nodes that will load data to the monitoring solution. It should produce data for CPU, RAM, Disk, etc.
* (T403, 3 pts) Capture the containers metric of Docker *(either via agent or by configuring the daemon)*
* (T404, 1 pts) One visualization that shows the CPU load of the monitored hosts, based on the collected information
* (T405, 1 pts) One visualization that shows the RAM utilization of the monitored hosts, based on the collected information
* (T406, 1 pts) One visualization that shows the number of containers *(any state – running, paused, etc.)* on the Docker host, based on the collected information
* (T407, 2 pts) A dashboard that includes all three visualizations

#### Containers (8 pts)

*You are expected to demonstrate knowledge working with* ***Docker****. On the picture it is displayed as* ***Do***

* (T501, 2 pts) Create a dedicated network (**app-net**) for the containers *(either as part of the pipeline or not)*
* (T502, 3 pts) Run and attach the containers (**Co1**, **Co2**, and **Co3**) to the dedicated network
* (T503, 3 pts) Working containerized application in production mode as expected *(for a valid and expected output, check the repository you cloned earlier)*