Wordpress on AWS demo

Purpose

Learn (or refresh) knowledge on some moderns DevOps tools on a real task.

The result of the task execution

- Running latest Ubuntu 20.04 instance with apache/php/mariadb, Wordpress site
- DNS name registered
- SSL certificate obtained.
- All done as automatically as possible.
- All with as latest stable versions as possible

This Project cover some use

- Jenkins 2.234 (pipelines)
- Docker 18.09.8 (Dockerfile, Docker Hub)
- Ubuntu 20.04 (AWS EC2 instance)
- Apache 2.4.41
- PHP7.4
- MariaDB 10.3.22
- terraform 0.12.24 (with AWS S3 state storage)
- ansible 2.7.17
- github (git repositories)
- GoDaddy API (Automated DNS update)
- AWS cli 1.18.48
- certbot 0.40.0
- wordpress 5.4.1
- nginx 1.17.9 (SSL offload reverse proxy for Jenkins)

Tools

Tool	Purpose	Related links
github	git code repository	https://github.com/kyivtank/jenkins https://github.com/kyivtank/socioniks.club
Docker	containers, Dockerfile	
Docker Hub	Store custom containers, auto build	https://hub.docker.com/repository/docker/kyivtank/jenkins
Jenkins	Pipelines, Automate build with github code changes	https://build.liutyi.info/job/socioniks.club/job/master/
terraform	AWS resource provisioning	
Ansible	Web server provisioning	
AWS	EC2 instance, S3 bucket	
GoDaddy API	Domain records update using API	
Letsencrypt	Free SSL certificate, using certbot	https://www.ssllabs.com/ssltest/analyze.html?d=socioniks.club
Wordpress	Setup wordpress with apache, php7, mariadb, wordpress-cli	https://socioniks.club/
nginx	(Optional) ssl offload loadbalancer for Jenkins	

Jenkins container

First, we need Jenkins with some plugins and dependencies (aws-cli, ansible,..). Using a docker container for that.

https://github.com/kyivtank/jenkins/blob/master/Dockerfile

https://github.com/kyivtank/jenkins/blob/master/plugins.txt

Dockerfile

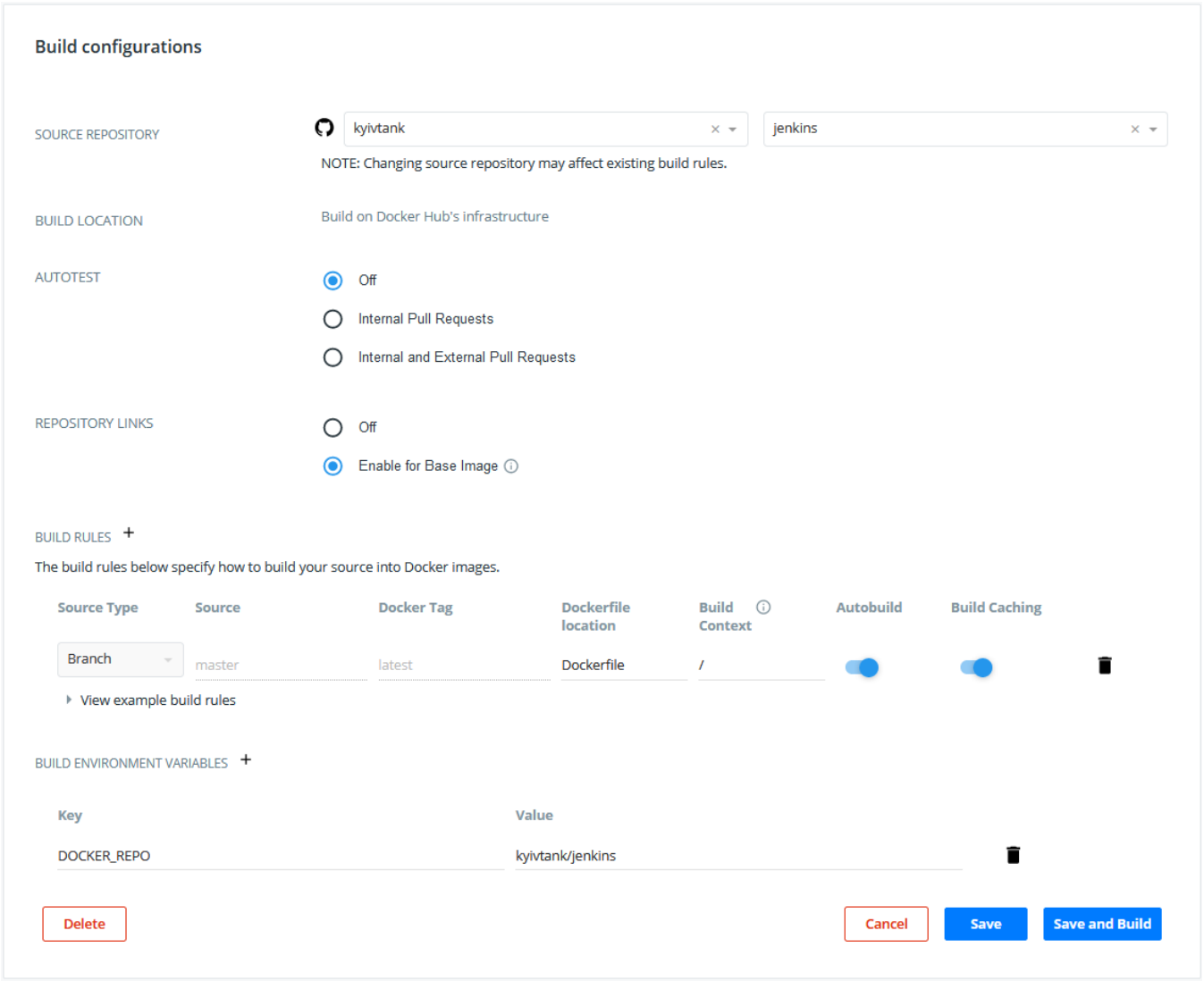
```
FROM jenkins/jenkins:alpine

# Skip initial setup
ENV JAVA_OPTS -Djenkins.install.runSetupWizard=false

#Pre-install plugins
COPY plugins.txt /usr/share/jenkins/plugins.txt
RUN /usr/local/bin/install-plugins.sh < /usr/share/jenkins/plugins.txt

# setup docker, docker-compose, terraform, ansible, jq
USER root
RUN apk add \
  docker \
  jq \
  ansible \
  python-dev \
  libffi-dev \
  openssl-dev \
  gcc \
  libc-dev \
  make \
  py-pip &&\
 pip install \
  docker-compose \
  awscli &&\
 wget https://releases.hashicorp.com/terraform/0.12.24/terraform_0.12.24_linux_amd64.zip &&\
 unzip terraform_0.12.24_linux_amd64.zip &&\
 mv terraform /usr/local/bin

USER jenkins
```

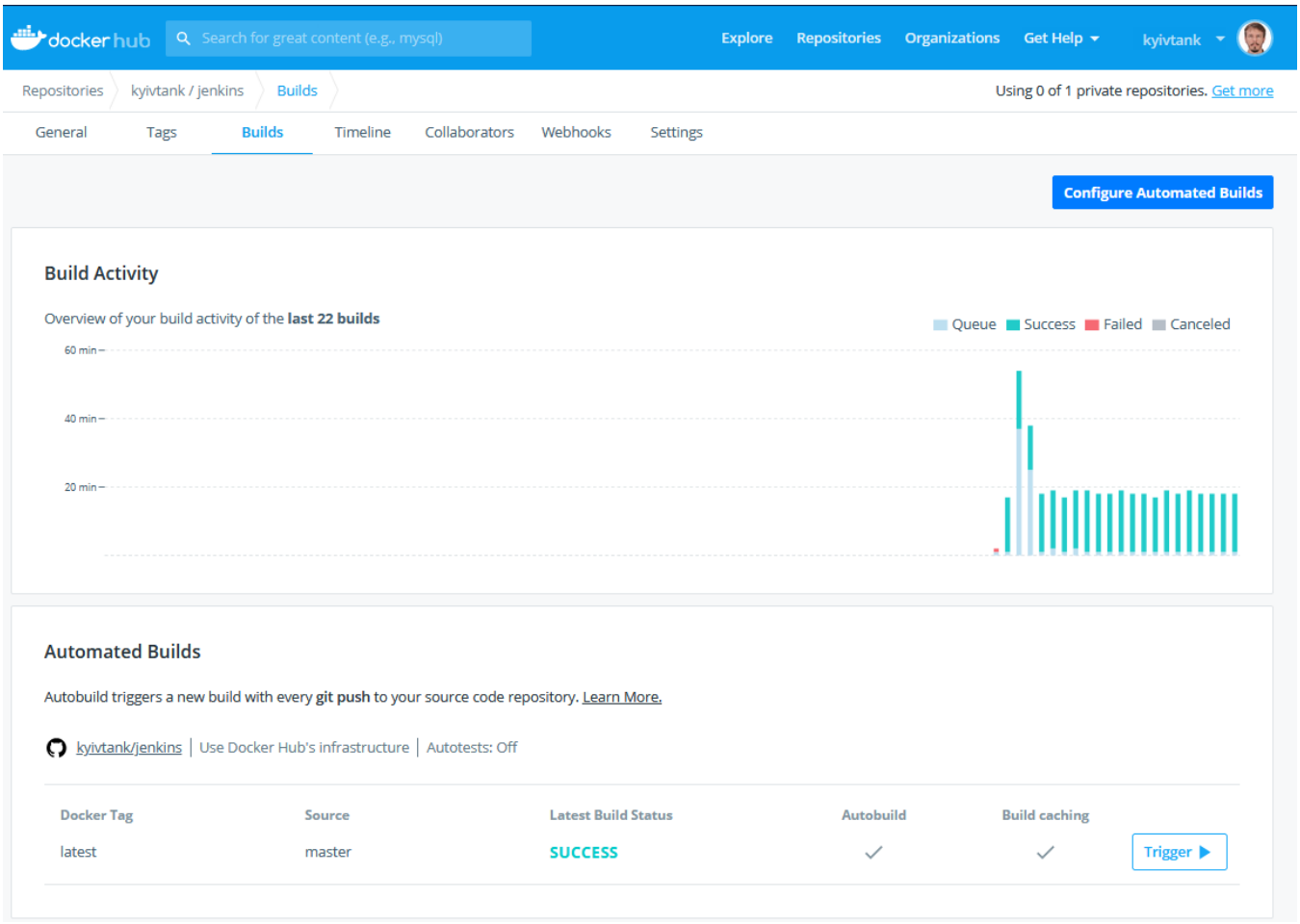
local build

```
docker build -t kyivtank/jenkins .
```

Docker Hub

To build and store container - docker hub with auto-build feature enabled. So every change in Dockerfile or base image will create an updated custom Jenkins image





Terraform EC2 instance setup

simple terraform profile for t2.micro ubuntu 20.04 instance with a security group that allows ports 22, 80, and 443.

Required:

- AWS account with API keys generated (Free tire is OK for this)
- S3 bucket (in the example it is "kyivtank-state") created.
- aws2020 ssh key created in AWS account
- Access keys must be available as environment variables.

terraform.tfvars

```
instance_type = "t2.micro"
keyname = "aws2020-key"
```

```
variable "instance_type" {
}

variable "keyname" {
}

terraform {
  backend "s3" {
    bucket = "kyivtank-state"
    key    = "terraform/terraform.tfstate"
    region = "eu-central-1"
  }
```

```
}

provider "aws" {
}

data "aws_ami" "ubuntu" {
    most_recent = true

    filter {
        name   = "name"
        values = ["ubuntu/images/hvm-ssd/ubuntu-focal-20.04-amd64-server-*"]
    }

    filter {
        name   = "virtualization-type"
        values = ["hvm"]
    }

    owners = ["099720109477"] # Canonical
}

resource "aws_instance" "web" {
    ami           = data.aws_ami.ubuntu.id
    instance_type   = var.instance_type
    key_name        = var.keyname
    vpc_security_group_ids = [aws_security_group.sg_allow_ssh_web.id]
    associate_public_ip_address = true
    tags = {
      Name = "Wordpress"
    }
}

resource "aws_security_group" "sg_allow_ssh_web" {
  name        = "allow_ssh_web"
  description = "Allow SSH and Web inbound traffic"

  ingress {
    from_port   = 22
    to_port     = 22
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 80
    to_port     = 80
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  ingress {
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    cidr_blocks = ["0.0.0.0/0"]
  }

  egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
  }
}

output "PUBLIC_IP" {
  value = aws_instance.web.public_ip
}
output "INSTANCE_ID" {
  value = aws_instance.web.id
```

```
}
```

local run of terraform

```
terraform init
terraform apply
terraform destroy
```

Setup Jenkins to run pipeline with terraform and ansible

1. start docker container with Jenkins
2. add credentials for gihub, godaddy, aws
3. setup multibranch pipeline job with repo that contains Jenkinsfile
4. Wait for the job to be executed

Github repo - https://github.com/kyivtank/socioniks.club

Jenkins build - https://build.liutyi.info/job/socionics.club/job/master/

Jenkins start

Start Jenkins (assuming you already have host with Docker installed)

```
#docker run -d -p 8080:8080 kyivtank/jenkins
docker run -d -p 8080:8080 -p 50000:50000 -v /docker/var/jenkins_home:/var/jenkins_home --restart=always
kyivtank/jenkins
```

Nginx SSL offload proxy for Jenkins (optional)

/etc/nginx/sites-enabled/build

```
upstream jenkins {
        server 127.0.0.1:8080 fail_timeout=0;
}
server {
        listen       80;
        server_name build.liutyi.info;
        return 301 https://$host$request_uri;
    }
server {
        listen 443 ssl http2;
        server_name build.liutyi.info;

        ssl on;
        ssl_certificate /etc/ssl/private/liutyi.info-wildcard-full.pem;
        ssl_certificate_key /etc/ssl/private/liutyi.info-wildcard.key;

        ssl_protocols TLSv1 TLSv1.1 TLSv1.2;
        ssl_ciphers 'HIGH:!aNULL:!MD5:!kEDH';
        ssl_session_cache shared:SSL:10m;
        ssl_session_timeout 5m;
        ssl_prefer_server_ciphers on;
        ssl_stapling on;
        ssl_stapling_verify on;
        ssl_trusted_certificate "/etc/ssl/certs/ca-certs.pem";
        resolver 8.8.8.8 8.8.4.4 valid=300s;
        resolver_timeout 5s;

       location / {
        proxy_set_header        Host $host:$server_port;
        proxy_set_header        X-Real-IP $remote_addr;
        proxy_set_header        X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header        X-Forwarded-Proto $scheme;
        proxy_redirect http:// https://;
        proxy_pass              http://jenkins;
        # Required for new HTTP-based CLI
        proxy_http_version 1.1;
        proxy_request_buffering off;
        proxy_buffering off; # Required for HTTP-based CLI to work over SSL
      }

}
```

Jenkins Pipeline script

Jenkinsfile

```
pipeline {
    agent any
       triggers {
        pollSCM "* * * * *"
       }
    environment {
        AWS_ACCESS_KEY_ID     = credentials('jenkins-aws-secret-key-id')
        AWS_SECRET_ACCESS_KEY = credentials('jenkins-aws-secret-access-key')
        AWS_DEFAULT_REGION = 'eu-central-1'
        AWS_DEFAULT_OUTPUT = 'table'
        GODADDY_DOMAIN = 'socioniks.club'
        GODADDY_API_KEY = credentials ('jenkins-godaddy-key')
        GODADDY_API_SECRET = credentials('jenkins-godaddy-secret')
        ANSIBLE_HOST_KEY_CHECKING = 'false'
        ANSIBLE_FORCE_COLOR = 'true'
    }
    stages {
        stage('Terraform') {
            steps {
                echo '=== AWS EC2 ==='
                dir("${env.WORKSPACE}/terraform") {
                 wrap([$class: 'AnsiColorBuildWrapper', 'colorMapName': 'xterm']) {
                  sh "terraform init"
                  sh "terraform apply -auto-approve"
                 }
                 sh 'aws ec2 wait instance-running --instance-ids `terraform output INSTANCE_ID`'
                 sh 'terraform output PUBLIC_IP > terraform.ip'
                }
            }
        }
        stage('GoDaddy') {
            steps {
                echo '=== DNS A record update ==='
                wrap([$class: 'AnsiColorBuildWrapper', 'colorMapName': 'xterm']) {
                 sh 'scripts/update_godaddy_dns.sh'
                }
            }
        }
        stage('Ansible') {
            steps {
                echo '=== Try ssh ==='
                wrap([$class: 'AnsiColorBuildWrapper', 'colorMapName': 'xterm']) {
                 withCredentials(bindings: [sshUserPrivateKey(credentialsId: 'aws2020', keyFileVariable:
'private_key')]){
                  sh "scripts/deploy_all.sh"
                 }
                }
            }
        }
    }
}
```

Jenkins Credentials

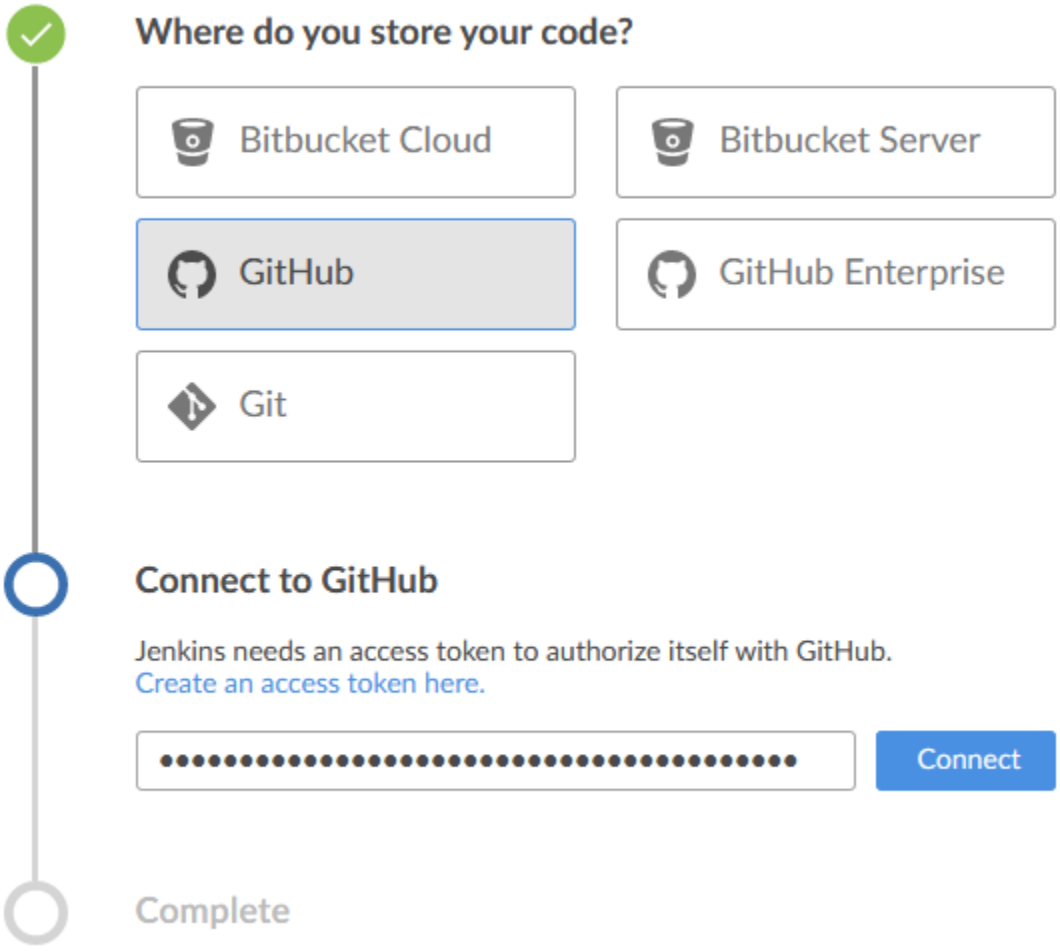
Global credentials (unrestricted)

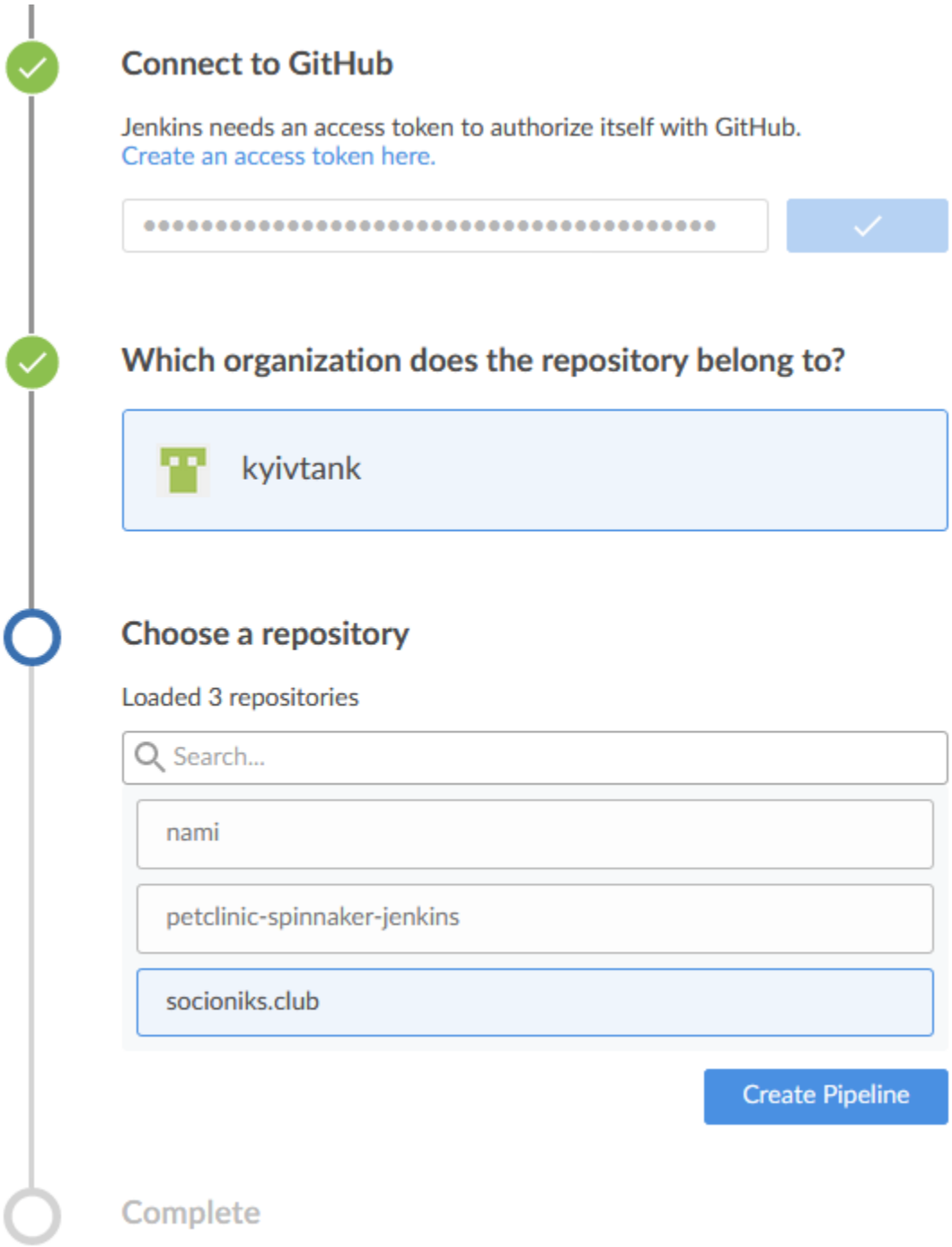
Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind
jenkins-godaddy-secret	jenkins-godaddy-secret	Secret text
jenkins-godaddy-key	jenkins-godaddy-key	Secret text
jenkins-aws-secret-access-key	jenkins-aws-secret-access-key	Secret text
jenkins-aws-secret-key-id	jenkins-aws-secret-key-id	Secret text
aws2020	ubuntu	SSH Username with private key

Jenkins Pipeline configuration

using Blue Ocean interface





Let's Encrypt certificate

manual test run

```
export GODADDY_DOMAIN=socioniks.club
certbot certonly --standalone --noninteractive --agree-tos --email noc@$GODADDY_DOMAIN -d $GODADDY_DOMAIN"
```

codebase ansible playbook

https://github.com/kyivtank/socioniks.club/blob/master/ansible/group_vars/all

ansible/group_vars/all

```
wp_apache: apache2
wp_admin_email: noc@socioniks.club
wp_site_url: socioniks.club
```

https://github.com/kyivtank/socioniks.club/blob/master/ansible/roles/letsencrypt/tasks/main.yml

ansible/roles/letsencrypt/tasks/main.yml

```
---
- name: restart apache
  service: name={{ wp_apache }} state=stopped

- name: Get Let's Encrypt Certificates
  shell: "certbot certonly --standalone --noninteractive --agree-tos --email {{ wp_admin_email }} -d {{
wp_site_url }}"
  args:
    creates: /etc/letsencrypt/live/{{ wp_site_url }}

- name: Schedule SSL certificate renewal
  cron:
    name: SSL Cert Renewal
    minute: 0
    hour: 20
    day: '*/10'
    user: root
    job: "/usr/bin/certbot renew"
```

GoDaddy API

manual test run

```
export GODADDY_API_KEY=dKxxxxxxxxx
export GODADDY_API_SECRET=8Exxxxxxxx
export GODADDY_DOMAIN=socioniks.club
IP=`terraform output INSTANCE_ID`
echo "Changing IP to $IP for domain $GODADDY_DOMAIN"
curl --silent -X PUT "https://api.godaddy.com/v1/domains/$GODADDY_DOMAIN/records/A/@" -H "accept: application
/json" -H "Content-Type: application/json" -H "Authorization: sso-key $GODADDY_API_KEY:$GODADDY_API_SECRET" -d
"[ { \"data\": \"$IP\", \"port\": 1, \"priority\": 1, \"protocol\": \"string\", \"service\": \"string\", \"
ttl\": 3600, \"weight\": 1 }]"
```

codebase script

https://github.com/kyivtank/socioniks.club/blob/master/scripts/update_godaddy_dns.sh

Manual add CAA

since GoDaddy API not yet support CAA, this line may be added manually

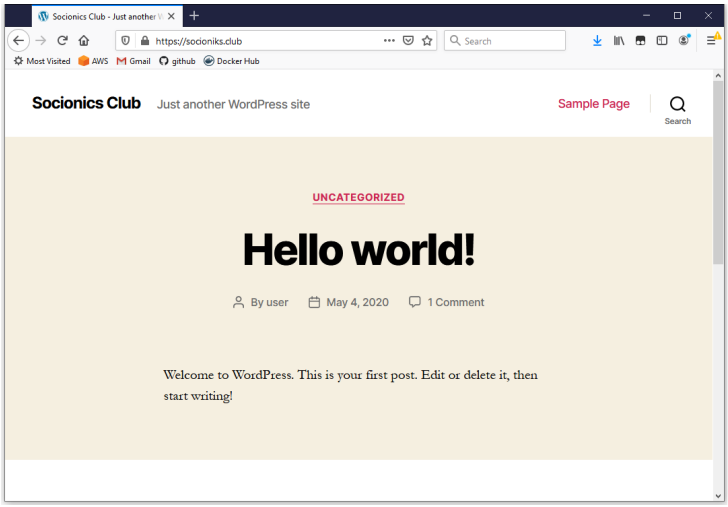
CAA	@	letsencrypt.org 128 issue	1 Hour	

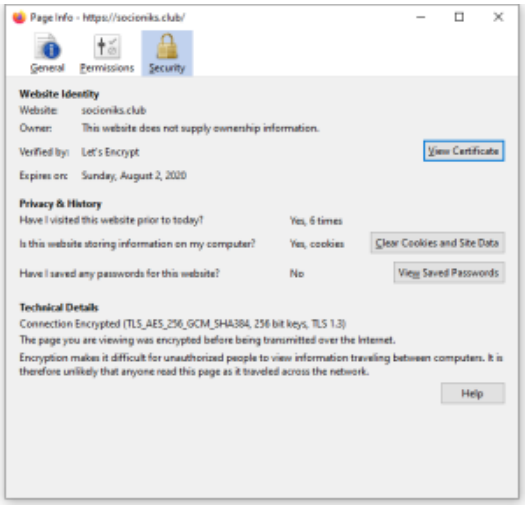
WP-CLI

Manual add user example

```
sudo wp user create liutyi noc@socioniks.club --first_name=Oleksandr --last_name=Liutyi --role=administrator --user_pass=password --allow-root --path=/var/www/html
```

Result





SSL Check