
QA
מע

אלכס גורבצ'וב
АЛЕКСЕЙ ГОРБАЧЁВ



JENKINS, CICD

JENKINS, CICD



Jenkins, CI/CD



Что такое Jenkins?

Jenkins - это инструмент для непрерывной интеграции и непрерывной доставки (CI/CD), который позволяет автоматизировать процессы сборки, тестирования и доставки программного обеспечения.

Jenkins позволяет создавать и запускать автоматические сборки приложений, тестировать их на разных платформах и конфигурациях, и доставлять готовое приложение на целевые серверы или платформы. **Jenkins** является открытым и бесплатным инструментом с большим сообществом пользователей и разработчиков, которые создают дополнительные плагины и интеграции для **Jenkins**.

Jenkins имеет графический интерфейс пользователя и удобную консоль управления, а также интегрируется с другими инструментами для автоматизации разработки, такими как Git, SVN, JIRA, Docker и многими другими.

<https://www.jenkins.io/>

Jenkins, CI/CD



Continuous integration

Continuous integration - Непрерывная интеграция

Практика включает в себя:

- Хранение кода в системе контроля версий (VCS)
- Code Review
- Частые автоматизированные сборки
- Модульное тестирование
- Статистическая проверка качества кода
- Автоматизированное развертывание на среды разработки
- Проверка работоспособности системы после развертывания

Основные инструменты CI:

- Git, Git-LFS, BitBucket
- JUnit, Jest
- Maven, Npm (yarn)
- SonarQube
- Sonatype Nexus OSS, Harbor
- Jenkins

Jenkins, CI/CD



Continuous delivery

Continuous delivery - Непрерывная поставка

Практика включает в себя:

- Хранение инсталляционного пакета в централизованном хранилище.
- Автоматизацию развертывания приложения на среды тестирования.
- Автоматические проверки успешности процесса развертывания.
- Интеграция с инструментами тестирования на безопасность.

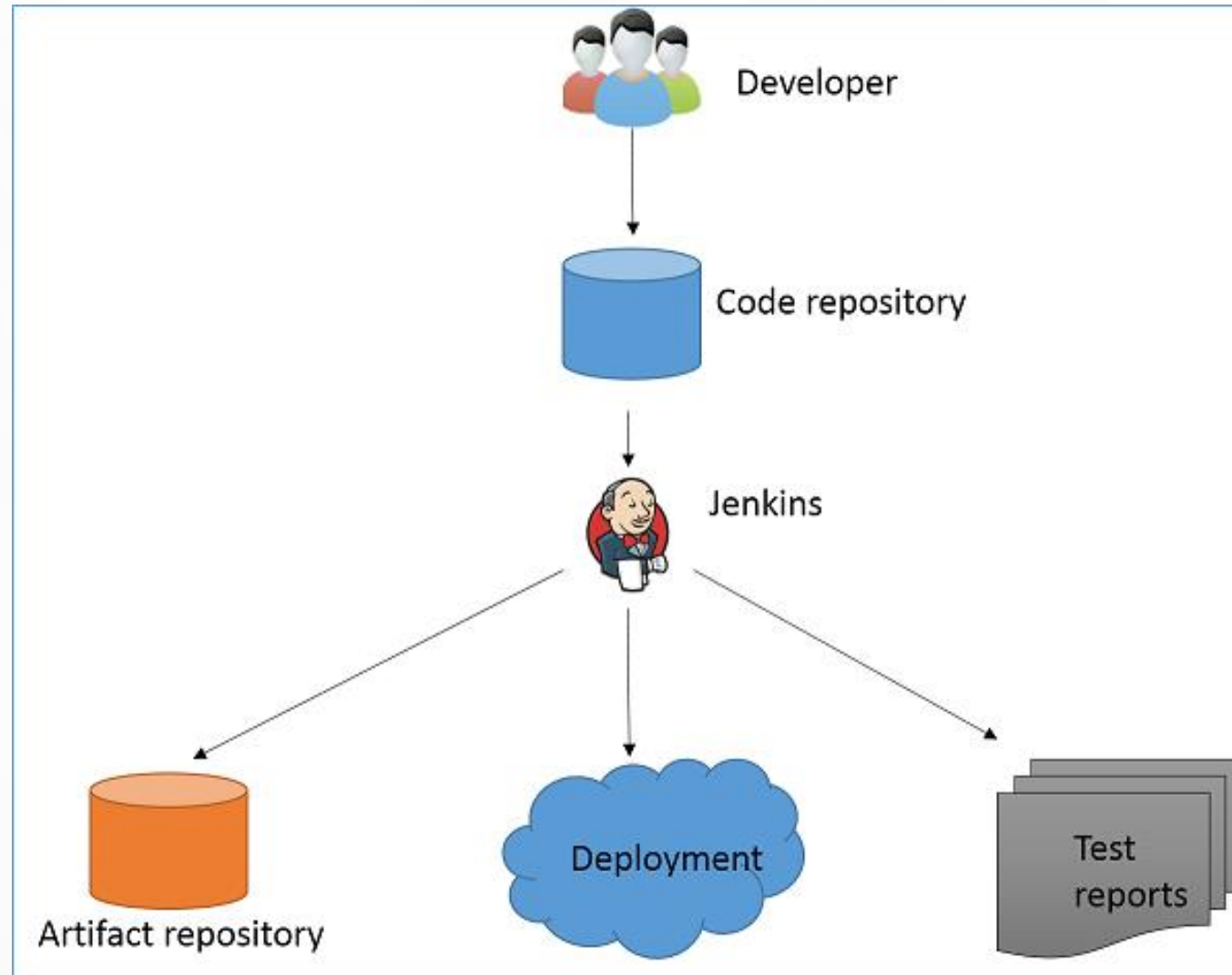
Основные инструменты CD:

- Sonatype Nexus OSS, Harbor
- Jenkins
- BitBucket
- Разворачиваются сервисы в кластере Kubernetes
- Проверка успешности развертывания осуществляется несколькими способами: Jenkins как основной оркестратор, средства мониторинга и работы с кластером Kibana, Rancher, Grafana

Jenkins, CI/CD



Jenkins workflow



Jenkins, CI/CD



Почему Jenkins?

- Открытый и бесплатный: **Jenkins** является бесплатным и открытым инструментом, что позволяет широко использовать его в индустрии.
- Легко настраивается и настраивается: **Jenkins** имеет гибкую конфигурацию и множество плагинов, что делает его очень настраиваемым и позволяет адаптировать его к различным потребностям и проектам.
- Автоматическая сборка и тестирование: **Jenkins** позволяет автоматически создавать сборки приложений и тестировать их на разных платформах и конфигурациях, что существенно ускоряет процесс разработки и снижает вероятность ошибок.
- Интеграция с другими инструментами: **Jenkins** может легко интегрироваться с другими инструментами для автоматизации разработки, такими как Git, SVN, JIRA, Docker и многими другими.
- Масштабируемость: **Jenkins** легко масштабируется, что позволяет использовать его как для небольших проектов, так и для крупных и сложных проектов с большим количеством компонентов.

Jenkins, CI/CD



Install Jenkins with Docker

- Create new Repository
- Push to GitHub
- Create new Dockerfile:

```
FROM jenkins/jenkins:2.375.1
USER root
RUN apt-get update && apt-get install -y lsb-release
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"
```

- Run: **docker build -t image_name.**

<https://www.jenkins.io/doc/book/installing/docker/>

<https://github.com/alexpitronot/Jenkins-Test>

Jenkins, CI/CD



Install Jenkins with Docker

- Build new network:
docker network create *network_name*
- Run container for Jenkins network:
docker run --name *container_name* **--rm --detach --privileged --network** *network_name* **--network-alias docker --env DOCKER_TLS_CERTDIR=/certs --volume jenkins-docker-certs:/certs/client --volume jenkins-data:/var/jenkins_home --publish 2376:2376 docker:dind --storage-driver overlay2**
- Run Jenkins:
docker run --name *container_name* **--rm --detach --network** *network_name* **--env DOCKER_HOST=tcp://docker:2376 --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 --publish 8080:8080 --publish 50000:50000 --volume jenkins-data:/var/jenkins_home --volume jenkins-docker-certs:/certs/client:ro** *image_name*

Jenkins, CI/CD



Install Jenkins with Docker

- Open:
http://localhost:8080

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (not sure where to find it?) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

Jenkins, CI/CD



Install Jenkins with Docker

- To unlock Jenkins, run:
docker logs *container_name*

Jenkins initial setup is required. An admin user has been created and a password generated.

Please use the following password to proceed to installation:

f45e2e18dbfb470ab29f810e61ce9686

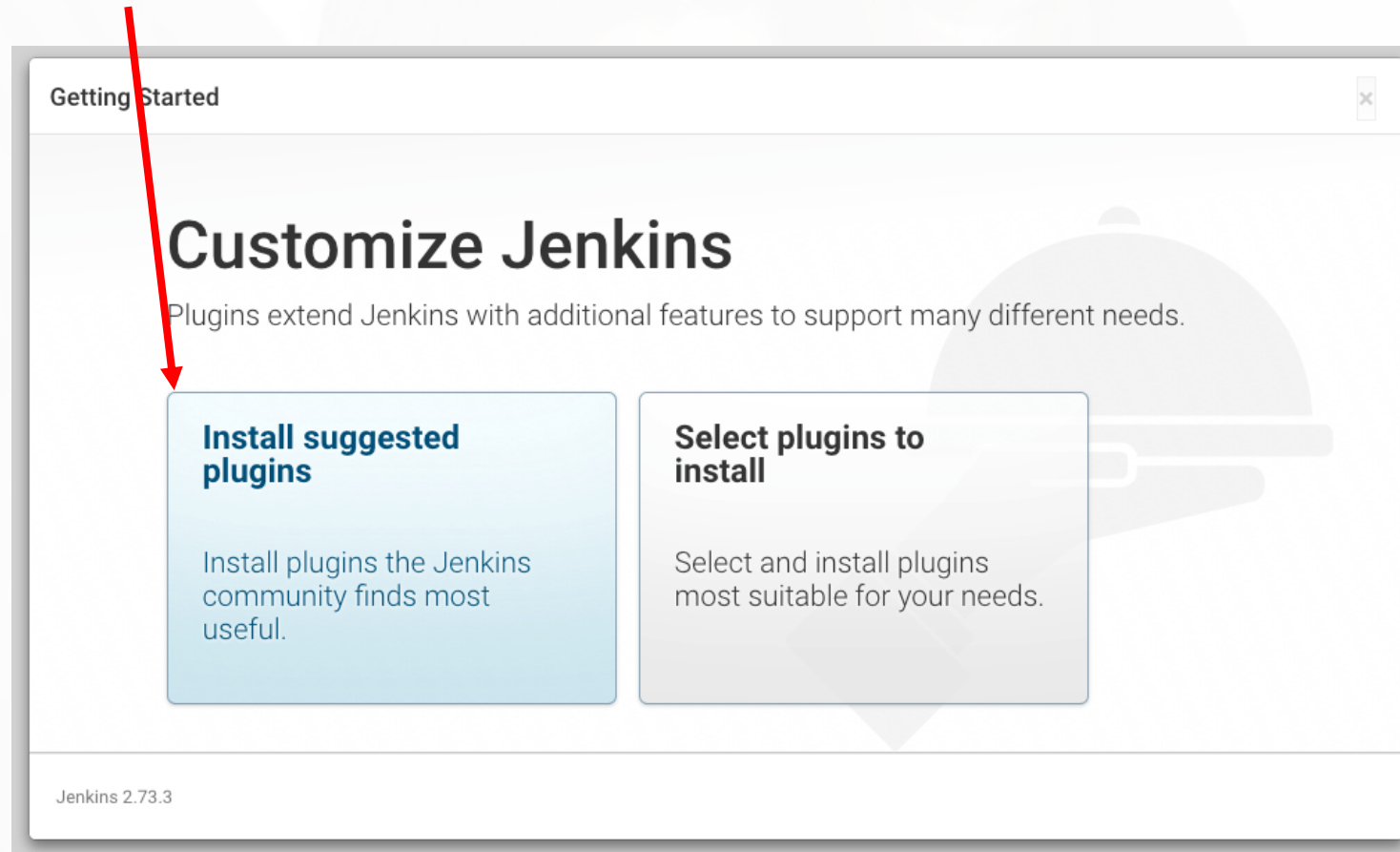
This may also be found at: `/var/jenkins_home/secrets/initialAdminPassword`

Jenkins, CI/CD



Install Jenkins with Docker

- Customizing Jenkins with plugins:



Jenkins, CI/CD



Install Jenkins with Docker

- Create first Admin:

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

[Take Screenshot](#)

☒ Grab the whole screen

☐ Grab the current window

☐ Select area to grab

Grab after a delay of seconds

[Effects](#)

☐ Include pointer

☐ Include the window border

Apply effect:


Jenkins 2.194

[Continue as admin](#) [Save and Continue](#)

Jenkins, CI/CD





Welcome to Jenkins!


 Jenkins

search


?


 1


 admin


 log out


Dashboard


 New Item


 People


 Build History

 Manage Jenkins

 My Views

 Open Blue Ocean

 Lockable Resources

 New View

Build Queue

No builds in the queue.

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

add description

Jenkins, CI/CD



Creating first Jenkins job

- To create a new job, we have two options to click on:

A screenshot of the Jenkins web interface. The top navigation bar is black with the Jenkins logo and name on the left, and a search bar on the right. Below the navigation bar is a sidebar with a 'Dashboard' dropdown menu. The first item in the sidebar, 'New Item', is circled in red. The main content area on the right has a 'Welcome to Jenkins!' heading, followed by a paragraph explaining the page's purpose. Below this is a section titled 'Start building your software project', which contains a button labeled 'Create a job' that is also circled in red. The button has a right-pointing arrow next to it.

Jenkins search

Dashboard ▾

- New Item
- People
- Build History
- Manage Jenkins
- My Views
- Open Blue Ocean
- Lockable Resources
- New View

Welcome to Jenkins!

This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

Create a job →

Jenkins, CI/CD



Creating first Jenkins job

- We will use the **Freestyle project** option and we will give a name for our project:

Jenkins

Dashboard ▾

Enter an item name

» Required field

- Freestyle project**
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do
- Multi-configuration project**
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Bitbucket Team/Project**
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.
- Folder**
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have mul as they are in different folders.
- GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

Jenkins, CI/CD



Creating first Jenkins job

- General configuration:

This screenshot shows the 'General' configuration tab in Jenkins. At the top, there are tabs for 'General', 'Source Code Management', 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The 'General' tab is active. It features a 'Description' field with the text 'Build and run Python program'. Below this field are links for '[Plain text] [Preview](#) [Hide preview](#)'. Further down, there are three checkboxes: 'Discard old builds' (unchecked), 'GitHub project' (unchecked), and 'This build requires lockable resources' (unchecked). A blue question mark icon is visible to the right of these checkboxes.

- Source code management:

This screenshot shows the 'Source Code Management' configuration tab in Jenkins. The tabs at the top are 'General', 'Source Code Management' (active), 'Build Triggers', 'Build Environment', 'Build', and 'Post-build Actions'. The title 'Source Code Management' is displayed. Below the title, there are two radio button options: 'None' (selected) and 'Git' (unselected).

Jenkins, CI/CD



Creating first Jenkins job

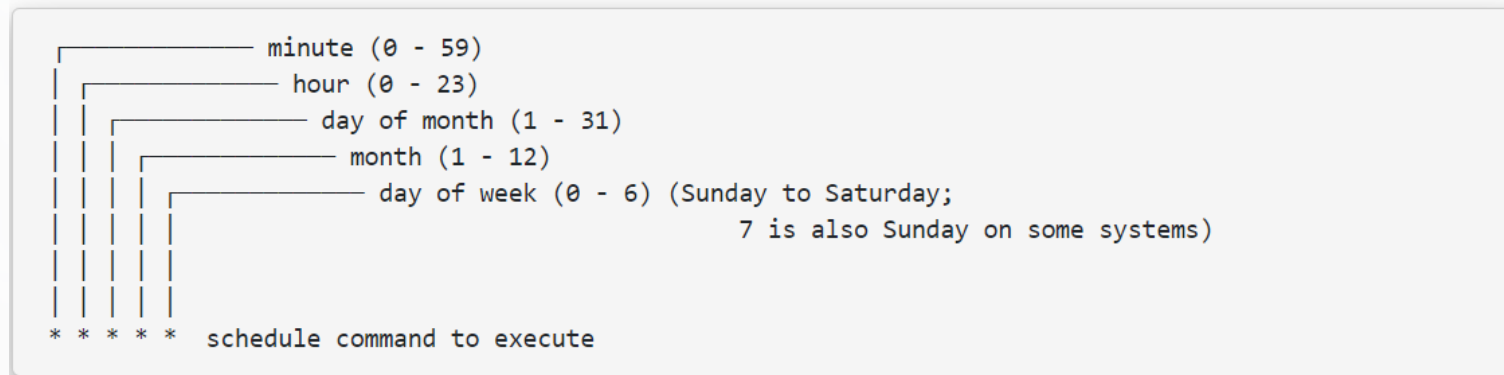
- Build triggers. I recommend to use H in every Jenkins schedule where it's possible to avoid spikes.
 - Build every hour: H * * * *
 - Build every 20 minutes: H/20 * * * *
 - Build every 20 minutes 5am to 11pm: H/20 5-23 * * *
 - Build every 20 minutes, work time/days (8am-6pm, MON-FRI) only: H/20 8-18 * * 1-5
 - Build every hour MON-WED and FRI only: H * * * 1-3,5
 - Build every hour, weekends in April and December: H * * 4,12 *
 - Build at 8.30am on July 4: 30 8 4 7 *

Jenkins, CI/CD



Creating first Jenkins job

- Jenkins schedule format:



- Build triggers. We will declare it to run every minute, there is a known bug in Jenkins for H/1:

Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☐ Build after other projects are built ?

☒ Build periodically ?

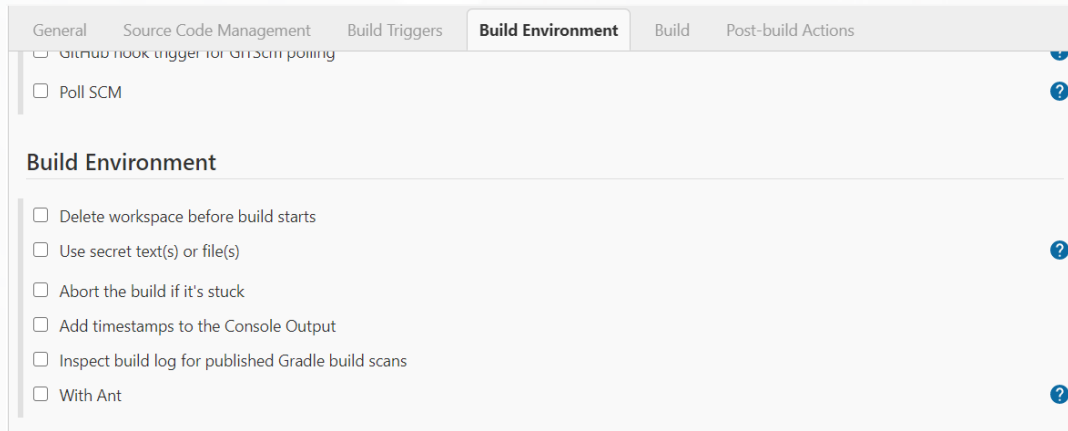
Schedule ?

Jenkins, CI/CD



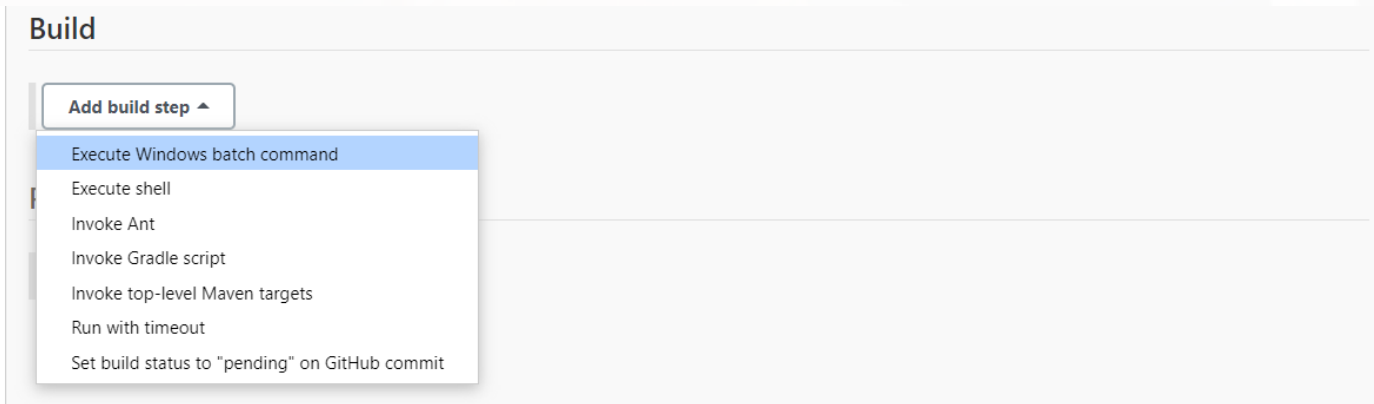
Creating first Jenkins job

- Build Environment:



The screenshot shows the 'Build Environment' tab in the Jenkins job configuration. The tabs at the top are General, Source Code Management, Build Triggers, Build Environment (selected), Build, and Post-build Actions. Under 'Build Triggers', there is a checkbox for 'Poll SCM'. Under 'Build Environment', there are several checkboxes: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Abort the build if it's stuck', 'Add timestamps to the Console Output', 'Inspect build log for published Gradle build scans', and 'With Ant'. Each checkbox has a blue question mark icon to its right.

- Build. Execute the Windows batch command:



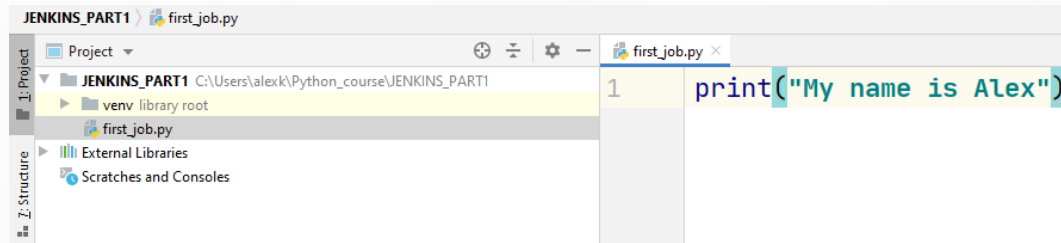
The screenshot shows the 'Build' tab in the Jenkins job configuration. A button labeled 'Add build step' is visible, and its dropdown menu is open, showing several options: 'Execute Windows batch command' (highlighted in blue), 'Execute shell', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'.

Jenkins, CI/CD



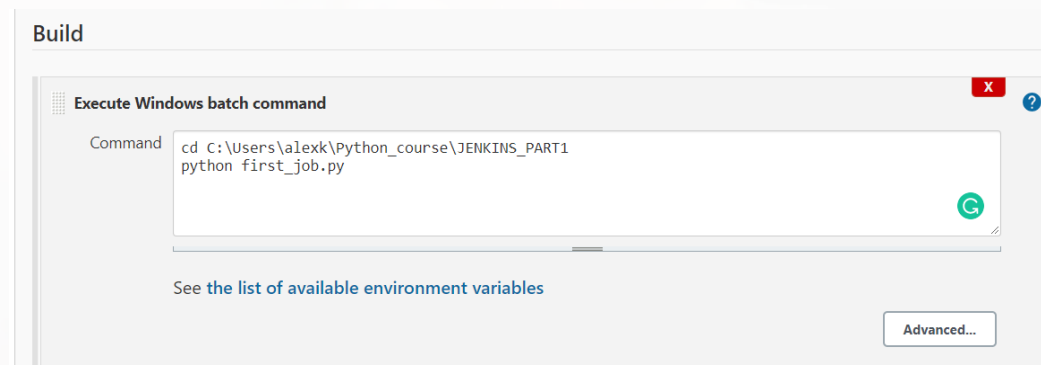
Creating first Jenkins job

- Let's create a simple Python file. We can see that we can execute the file from our terminal:



```
JENKINS_PART1 > first_job.py
Project
  JENKINS_PART1 C:\Users\alexx\Python_course\JENKINS_PART1
    venv library root
    first_job.py
  External Libraries
  Scratches and Consoles
1 print("My name is Alex")
```

- We will implement the same commands in our build:



Jenkins, CI/CD



Creating first Jenkins job

- Build Environment:

The screenshot shows the Jenkins configuration page for a new job, with the 'Build Environment' tab selected. The 'General' tab is also visible. The 'Build Environment' section contains several checkboxes: 'Delete workspace before build starts', 'Use secret text(s) or file(s)', 'Abort the build if it's stuck', 'Add timestamps to the Console Output', 'Inspect build log for published Gradle build scans', and 'With Ant'. The 'With Ant' checkbox is currently checked. There are blue question mark icons next to the 'Poll SCM', 'Use secret text(s) or file(s)', and 'With Ant' options.

- Build. Execute the Windows batch command:

The screenshot shows the Jenkins configuration page for a new job, with the 'Build' tab selected. The 'Add build step' button is highlighted, and a dropdown menu is open showing various build steps. The 'Execute shell' option is selected and highlighted with a red border. Other options in the dropdown include 'Execute Windows batch command', 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'.

Jenkins, CI/CD



Creating first Jenkins job

- Let's create a simple Python file. We can see that we can execute the file from our terminal:

```
test.py
1 print("Hello World")
2 print("Hello World again")
3
```

- We will implement the same commands in our build:

Build

Execute shell

Command

```
python3 test.py
```

[See the list of available environment variables](#)

Jenkins, CI/CD



Remote build trigger

- We can build our job to be triggered remotely:

General Source Code Management **Build Triggers** Build Environment Build Post-build Actions

Build Triggers

☒ Trigger builds remotely (e.g., from scripts) ?

Authentication Token

Use the following URL to trigger build remotely: JENKINS_URL/job/HelloWorld/build?token=TOKEN_NAME or /buildWithParameters?token=TOKEN_NAME

Optionally append &cause=Cause+Text to provide text that will be included in the recorded build cause.

☐ Build after other projects are built ?

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

- Now we will build our URL to trigger the build remotely:

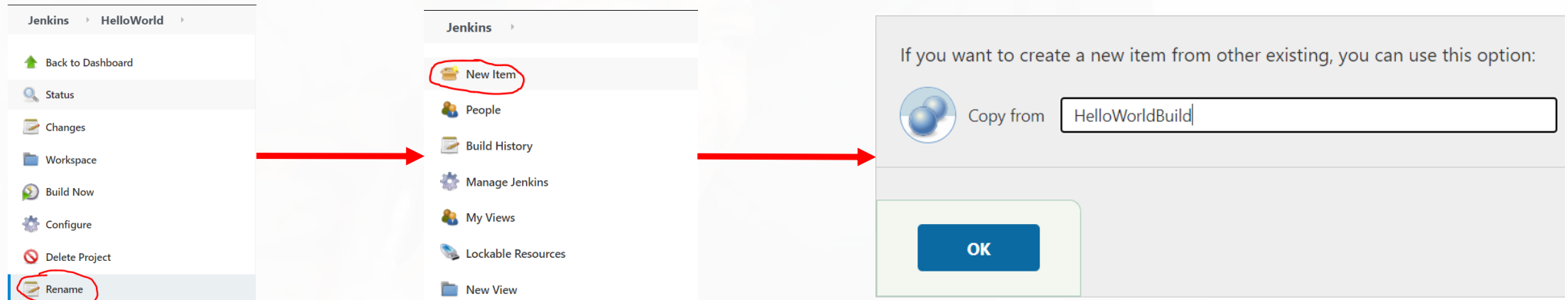
`http://localhost:8082/job/Hello/build?token=12345`

Jenkins, CI/CD



Job chaining in Jenkins

- Let's separate our current Job into three different jobs. One job will build, one job will run, and one job will display a relevant message.



Jenkins, CI/CD



Job chaining in Jenkins

- Let's configure the run job:

Build Triggers

☐ Trigger builds remotely (e.g., from scripts) ?

☒ Build after other projects are built ?

Projects to watch

☒ Trigger only if build is stable

☐ Trigger even if the build is unstable

☐ Trigger even if the build fails

☐ Build periodically ?

☐ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

Build

Execute Windows batch command [X] ?

Command

```
echo "-----"  
echo "A building process"  
echo "-----"
```

Jenkins, CI/CD



Job chaining in Jenkins

- Let's create the display job:

Build

Execute Windows batch command

Command

```
echo "-----"
echo "A building process"
echo "-----"
```

Build

Execute Windows batch command

Command

```
echo "-----"
echo "Success!!!"
echo "-----"
```

[See the list of available environment variables](#)

Advanced...

Jenkins, CI/CD



Job chaining in Jenkins

- Let's start the HelloWorldBuild job and watch the chain of the other jobs:

All	+			Name ↓	Last Success	Last Failure	Last Duration	
S	W							
				HelloWorldBuild	43 min - #30	1 hr 11 min - #8	1.1 sec	
				HelloWorldDisplay	N/A	N/A	N/A	
				HelloWorldRun	N/A	N/A	N/A	

Build Queue (1) ^

[HelloWorldRun](#)

Build Queue (1) ^

[HelloWorldDisplay](#)

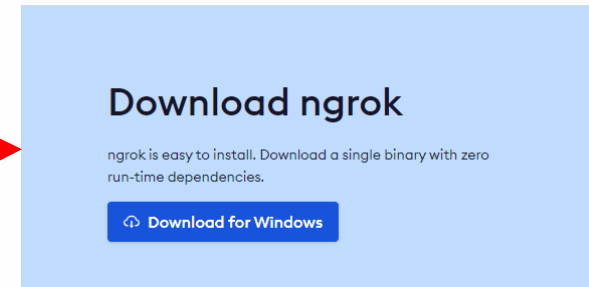
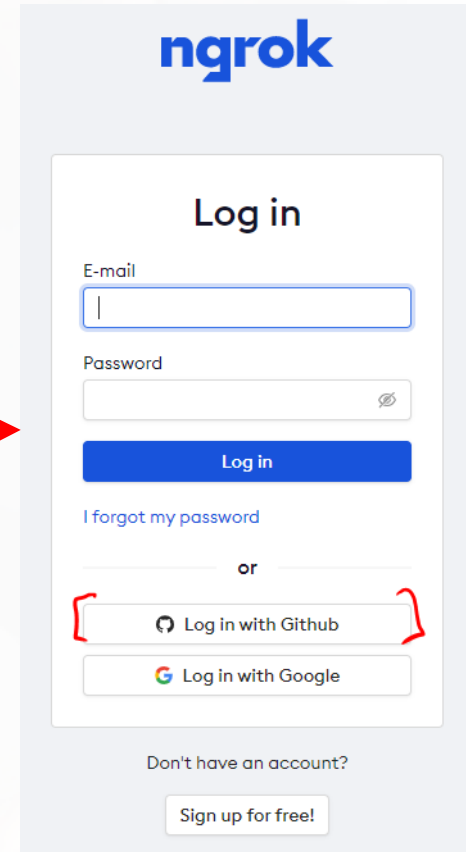
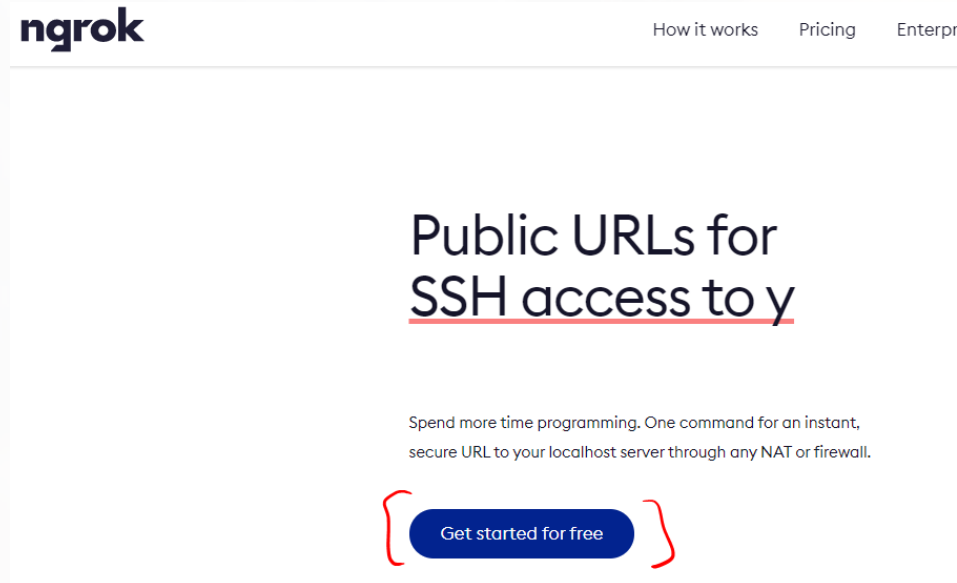
All	+			Name ↓	Last Success	Last Failure	Last Duration	
S	W							
				HelloWorldBuild	25 sec - #32	1 hr 21 min - #8	1.2 sec	
				HelloWorldDisplay	5.1 sec - #1	N/A	1 sec	
				HelloWorldRun	15 sec - #2	6 min 0 sec - #1	1.2 sec	

Jenkins, CI/CD



Configure (SCM) GitHub Triggers and Git Polling using Ngrok

- Install and Run Ngrok:
<https://ngrok.com/>



Jenkins, CI/CD



Configure (SCM) GitHub Triggers and Git Polling using Ngrok

- Connect your account:

The screenshot shows the Ngrok web interface. On the left is a dark sidebar with the 'ngrok' logo at the top. Below the logo are several menu items: 'Getting Started' (with a bell icon), 'Setup & Installation', 'Your Authtoken' (highlighted in blue), 'Tutorials', 'Endpoints' (with a globe icon), 'Tunnel Agents' (with a double arrow icon), 'TLS' (with a shield icon), 'Events' (with a lightning bolt icon), 'API' (with a plug icon), 'IP Policies' (with a location pin icon), 'Team' (with a person icon), 'Billing' (with a calendar icon), and 'Settings' (with a gear icon). The main content area has a light blue header with the title 'Your Authtoken'. Below the header, it says 'This is your personal Authtoken. Use this to authenticate the ngrok agent that you downloaded.' Below this text is a white box containing the auth token '1xRvzxi4RXGqWYOjGjiFIbTiuyT_51DQpiRZfi82hAjRSnJvX' and a blue 'Copy' button with a clipboard icon. At the bottom of the main area, there is a 'Command Line' section with the text 'Authenticate your ngrok agent. You only have to do this once. The Authtoken is saved in the default configuration file.' and a code block containing the command '\$./ngrok authtoken 1xRvzxi4RXGqWYOjGjiFIbTiuyT_51DQpiRZfi82hAjRSnJvX'.

Jenkins, CI/CD



Configure (SCM) GitHub Triggers and Git Polling using Ngrok

- Run `./ngrok http 8080` which will point to our Jenkins server:

```
VERSION:
2.3.40

inconshreveable - <alan@ngrok.com>
ngrok by @inconshreveable

Session Status      online
Account             alexpitrone (Plan: Free)
Version             2.3.40
Region              United States (us)
Web Interface       http://127.0.0.1:4040
Forwarding           http://4c9d-5-29-16-209.ngrok.io -> http://localhost:8080
Forwarding           https://4c9d-5-29-16-209.ngrok.io -> http://localhost:8080

Connections         ttl    opn    rt1    rt5    p50    p90
                   10     0      0.01   0.02   5.47   8.03

HTTP Requests
-----

GET /favicon.ico                200 OK
GET /static/060962e1/css/simple-page-variables.css 200 OK
GET /static/060962e1/images/jenkins.svg           200 OK
GET /static/060962e1/css/simple-page.theme.css     200 OK
GET /static/060962e1/css/simple-page-forms.css     200 OK
GET /login                                           200 OK
GET /static/060962e1/css/simple-page.css           200 OK
GET /                                                403 Forbidden
GET /static/060962e1/images/jenkins.svg           200 OK
GET /static/060962e1/css/simple-page-variables.css 200 OK
```

Jenkins, CI/CD



Configure (SCM) GitHub Triggers and Git Polling using Ngrok

- Setting up GitHub Webhook:

The screenshot shows the GitHub 'Webhooks / Manage webhook' page. Red arrows indicate the following steps:

- An arrow points from the 'Settings' tab in the top navigation bar to the 'Webhooks / Manage webhook' page.
- An arrow points from the 'Webhooks' option in the left sidebar to the 'Webhooks / Manage webhook' page.
- An arrow points from the 'Payload URL' input field to the 'Active' checkbox at the bottom.

The 'Webhooks / Manage webhook' page includes the following fields and options:

- Settings** (selected) and **Recent Deliveries** tabs.
- Text: "We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#)."
- Payload URL ***: `http://4c9d-5-29-16-209.ngrok.io/github-webhook/`
- Content type**: `application/x-www-form-urlencoded`
- Secret**: (empty input field)
- Which events would you like to trigger this webhook?**:
 - ☒ Just the push event.
 - ☐ Send me everything.
 - ☐ Let me select individual events.
- Active**: ☒ (checked). Below it: "We will deliver event details when this hook is triggered."
- Buttons**: **Update webhook** (green) and **Delete webhook** (red).

Jenkins, CI/CD



Configure (SCM) GitHub Triggers and Git Polling using Ngrok

- Create a new Jenkins Pipeline Job:

Dashboard > git-pipe >

General **Build Triggers** Advanced Project Options Pipeline

☐ Throttle builds ?

Build Triggers

☐ Build after other projects are built ?

☐ Build periodically ?

☒ GitHub hook trigger for GITScm polling ?

☐ Poll SCM ?

☐ Disable this project ?

☐ Quiet period ?

☐ Trigger builds remotely (e.g., from scripts) ?

Advanced Project Options

Advanced...

Pipeline

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/alexpitronot/hello-world.git

Credentials ?

alexpitronot/***** Add

Jenkins, CI/CD



Building Docker Images to Docker Hub Using Jenkins Pipelines

<https://dzone.com/articles/building-docker-images-to-docker-hub-using-jenkins>

<https://github.com/alexpitronot/far-2-cel/tree/master>



Thanks for your time 😊