StackStorm Installation Guide

**Quick Install on Ubuntu**

Grab a **clean** 64-bit Linux system that fits the [system requirements](https://docs.stackstorm.com/install/system_requirements.html). Make sure that curl is up to date using sudo apt-get install curl on Ubuntu, or sudo yum install curl nss on RHEL/CentOS. Then run this command:

curl -sSL https://stackstorm.com/packages/install.sh | bash -s -- --user**=**st2admin --password**=**'Ch@ngeMe'

This is an opinionated installation of StackStorm. It will download and install all components, as per the [single host reference deployment](https://docs.stackstorm.com/install/overview.html). It assumes that you have a clean, basic installation of Ubuntu or RHEL/CentOS.

If you are trying to install StackStorm on a server with other applications, or local customisations, you may run into problems. In that case, you should use one of the manual installation methods.

The script itself is not idempotent. If you try to re-run the script on top of a failed installation, it will fail. Start again with a clean system, or switch to a manual install.

# **Step by Step Installation and Configuration on Ubuntu Bionic (18.04)**

If you’re just looking for a quick “one-liner” installation, check the [top-level install guide](https://docs.stackstorm.com/install/index.html). If you need a customised installation, use this guide for step-by-step instructions for installing StackStorm on a single system as per the [Reference deployment](https://docs.stackstorm.com/install/overview.html).

StackStorm on Ubuntu 18.04 runs all services, actions and sensors using Python 3 **only**. It does not support Python2 actions.

Mistral is not supported on Ubuntu 18.04. All workflows must be written in [Orquesta](https://docs.stackstorm.com/orquesta/index.html).

## [**Minimal Installation**](https://docs.stackstorm.com/install/u18.html#id2)

### [**Install Dependencies**](https://docs.stackstorm.com/install/u18.html#id3)

Install MongoDB, and RabbitMQ:

sudo apt-get update

sudo apt-get install -y curl

*# Add key and repo for MongoDB (4.0)*

wget -qO - https://www.mongodb.org/static/pgp/server-4.0.asc | sudo apt-key add -

sudo sh -c "cat <<EOT > /etc/apt/sources.list.d/mongodb-org-4.0.list

deb http://repo.mongodb.org/apt/ubuntu **$(**lsb\_release -c | awk '{print $2}'**)**/mongodb-org/4.0 multiverse

EOT"

sudo apt-get update

sudo apt-get install -y crudini

sudo apt-get install -y mongodb-org

sudo apt-get install -y rabbitmq-server

For Ubuntu Bionic you may need to enable and start MongoDB.

sudo systemctl enable mongod

sudo systemctl start mongod

### [**Setup Repositories**](https://docs.stackstorm.com/install/u18.html#id4)

The following script will detect your platform and architecture and setup the appropriate StackStorm repository. It will also add the the GPG key used for package signing.

curl -s https://packagecloud.io/install/repositories/StackStorm/stable/script.deb.sh | sudo bash

### [**Install StackStorm Components**](https://docs.stackstorm.com/install/u18.html#id5)

sudo apt-get install -y st2

If you are not running RabbitMQ, MongoDB or PostgreSQL on the same system, or have changed the defaults, please adjust these settings:

* RabbitMQ connection at /etc/st2/st2.conf and /etc/mistral/mistral.conf
* MongoDB at /etc/st2/st2.conf
* PostgreSQL at /etc/mistral/mistral.conf

See the [Configuration documentation](https://docs.stackstorm.com/install/config/config.html) for more information.

### [**Setup Datastore Encryption**](https://docs.stackstorm.com/install/u18.html#id6)

The [Key-value store](https://docs.stackstorm.com/datastore.html) allows users to store encrypted values (secrets). These are stored using symmetric encryption (AES256). To generate a crypto key, run these commands:

DATASTORE\_ENCRYPTION\_KEYS\_DIRECTORY**=**"/etc/st2/keys"

DATASTORE\_ENCRYPTION\_KEY\_PATH**=**"${DATASTORE\_ENCRYPTION\_KEYS\_DIRECTORY}/datastore\_key.json"

sudo mkdir -p ${DATASTORE\_ENCRYPTION\_KEYS\_DIRECTORY}

sudo st2-generate-symmetric-crypto-key --key-path ${DATASTORE\_ENCRYPTION\_KEY\_PATH}

*# Make sure only st2 user can read the file*

sudo chgrp st2 ${DATASTORE\_ENCRYPTION\_KEYS\_DIRECTORY}

sudo chmod o-r ${DATASTORE\_ENCRYPTION\_KEYS\_DIRECTORY}

sudo chgrp st2 ${DATASTORE\_ENCRYPTION\_KEY\_PATH}

sudo chmod o-r ${DATASTORE\_ENCRYPTION\_KEY\_PATH}

*# set path to the key file in the config*

sudo crudini --set /etc/st2/st2.conf keyvalue encryption\_key\_path ${DATASTORE\_ENCRYPTION\_KEY\_PATH}

sudo st2ctl restart-component st2api

### [**Configure SSH and SUDO**](https://docs.stackstorm.com/install/u18.html#id7)

To run local and remote shell actions, StackStorm uses a special system user (by default stanley). For remote Linux actions, SSH is used. We recommend configuring public key-based SSH access on all remote hosts. We also recommend configuring SSH access to localhost for running examples and testing.

Create StackStorm system user, enable passwordless sudo, and set up ssh access to “localhost” so that SSH-based actions can be tested locally. You will need elevated privileges to do this:  
*# Create an SSH system user (default `stanley` user may already exist)*

sudo useradd stanley

sudo mkdir -p /home/stanley/.ssh

sudo chmod 0700 /home/stanley/.ssh

*# Generate ssh keys*

sudo ssh-keygen -f /home/stanley/.ssh/stanley\_rsa -P ""

*# Authorize key-based access*

sudo sh -c 'cat /home/stanley/.ssh/stanley\_rsa.pub >> /home/stanley/.ssh/authorized\_keys'

sudo chown -R stanley:stanley /home/stanley/.ssh

*# Enable passwordless sudo*

sudo sh -c 'echo "stanley ALL=(ALL) NOPASSWD: SETENV: ALL" >> /etc/sudoers.d/st2'

sudo chmod 0440 /etc/sudoers.d/st2

*# Make sure `Defaults requiretty` is disabled in `/etc/sudoers`*

sudo sed -i -r "s/^Defaults\s+\+?requiretty/# Defaults +requiretty/g" /etc/sudoers

* Configure SSH access and enable passwordless sudo on the remote hosts which StackStorm will be running remote actions on via SSH. Using the public key generated in the previous step, follow the instructions at [Configure SSH](https://docs.stackstorm.com/install/config/config.html#config-configure-ssh). To control Windows boxes, configure access for [Windows runners](https://docs.stackstorm.com/install/config/winrm_runners.html).

If you are using a different user, or path to their SSH key, you will need to change this section in /etc/st2/st2.conf:  
**[system\_user]**

user **=** stanley

ssh\_key\_file **=** /home/stanley/.ssh/stanley\_rsa

### [**Start Services**](https://docs.stackstorm.com/install/u18.html#id8)

Start services:  
sudo st2ctl start

Register sensors, rules and actions:  
sudo st2ctl reload

### [**Verify**](https://docs.stackstorm.com/install/u18.html#id9)

The following commands will test your StackStorm installation. They should all complete successfully:

st2 --version

st2 -h

*# List the actions from a 'core' pack*

st2 action list --pack**=**core

*# Run a local shell command*

st2 run core.local -- date -R

*# See the execution results*

st2 execution list

*# Fire a remote comand via SSH (Requires passwordless SSH)*

st2 run core.remote hosts**=**'localhost' -- uname -a

*# Install a pack*

st2 pack install st2

Use the supervisor script to manage StackStorm services:

sudo st2ctl start|stop|status|restart|restart-component|reload|clean

At this point you have a minimal working installation, and can happily play with StackStorm: follow the [Quick Start](https://docs.stackstorm.com/start.html) tutorial, [deploy the examples](https://docs.stackstorm.com/start.html#start-deploy-examples), explore and install packs from [StackStorm Exchange](https://exchange.stackstorm.org/).

But there is no joy without a Web UI, no security without SSL or authentication, no fun without ChatOps, and no money without Extreme Workflow Composer. Read on!

## [**Configure Authentication**](https://docs.stackstorm.com/install/u18.html#id10)

The reference deployment uses a file-based authentication provider for simplicity. Refer to [Authentication](https://docs.stackstorm.com/authentication.html) to configure and use PAM or LDAP authentication backends.

To set up authentication with file-based provider:

Create a user with a password:  
*# Install htpasswd utility if you don't have it*

sudo apt-get install -y apache2-utils

*# Create a user record in a password file.*

echo 'Ch@ngeMe' | sudo htpasswd -i /etc/st2/htpasswd st2admin

Enable and configure authentication in /etc/st2/st2.conf:  
**[auth]**

*# ...*

enable **=** True

backend **=** flat\_file

backend\_kwargs **=** {"file\_path": "/etc/st2/htpasswd"}

*# ...*

Restart the st2api service:  
sudo st2ctl restart-component st2api

Authenticate, and check that it works:  
*# Login - you will be prompted for password (default 'Ch@ngeMe')*

st2 login st2admin

*# Check that it works*

st2 action list

## [**Install WebUI and Setup SSL Termination**](https://docs.stackstorm.com/install/u18.html#id11)

[NGINX](http://nginx.org/) is used to serve WebUI static files, redirect HTTP to HTTPS, provide SSL termination, and reverse-proxy st2auth and st2api API endpoints. To set it up: install the st2web and nginx packages, generate certificates or place your existing certificates under /etc/ssl/st2, and configure nginx with StackStorm’s supplied [site config file st2.conf](https://github.com/StackStorm/st2/tree/master/conf/nginx/st2.conf).

*# Install st2web and nginx*

sudo apt-get install -y st2web nginx

*# Generate self-signed certificate or place your existing certificate under /etc/ssl/st2*

sudo mkdir -p /etc/ssl/st2

sudo openssl req -x509 -newkey rsa:2048 -keyout /etc/ssl/st2/st2.key -out /etc/ssl/st2/st2.crt \

-days XXX -nodes -subj "/C=US/ST=California/L=Palo Alto/O=StackStorm/OU=Information \

Technology/CN=**$(**hostname**)**"

*# Remove default site, if present*

sudo rm /etc/nginx/conf.d/default.conf

*# Copy and enable the supplied nginx config file*

sudo cp /usr/share/doc/st2/conf/nginx/st2.conf /etc/nginx/conf.d/

sudo service nginx restart

If you modify ports, or url paths in the nginx configuration, make the corresponding changes in the st2web configuration at /opt/stackstorm/static/webui/config.js.

Use your browser to connect to https://${ST2\_HOSTNAME} and login to the WebUI.

If you are trying to access the API from outside the box and you have configured nginx according to these instructions, use https://${EXTERNAL\_IP}/api/v1/${REST\_ENDPOINT}.

For example:

curl -X GET -H 'Connection: keep-alive' -H 'User-Agent: manual/curl' -H 'Accept-Encoding: gzip, deflate' -H 'Accept: \*/\*' -H 'X-Auth-Token: <YOUR\_TOKEN>' https://1.2.3.4/api/v1/actions

Similarly, you can connect to auth REST endpoints with https://${EXTERNAL\_IP}/auth/v1/${AUTH\_ENDPOINT}.

You can see the actual REST endpoint for a resource by adding a --debug option to the CLI command for the appropriate resource.

For example, to see the endpoint for getting actions, invoke:

st2 --debug action list

**Jira Server Installation on Ubuntu**

Please visit <https://linoxide.com/tools/install-jira-on-ubuntu-linux/> for detailed guide with pictures

## **Jira linux Installation Prerequisites**

**Before we get started with the installation, ensure you have the following:**

1. **An instance of** [**Ubuntu 18.04**](https://linoxide.com/distros/install-ubuntu-18-04-dual-boot-windows-10/) **/ Debian 10**
2. **Have** [**MySQL database installed**](https://linoxide.com/linux-how-to/install-mysql-ubuntu/) **on your instance. JIRA requires a database to install all the files so MySQL will come in handy.**

**Let's now dive in and get along with the installation.**

## **Step 1: Create a database for JIRA**

**JIRA software requires a database for its installation, therefore the first step will be to** [**create a database**](https://linoxide.com/tools/sql-commands-mariadb/) **in the MySQL database engine.**

**Login to MySQL as a root user**

**# mysql -u root -p**

**Create a database for JIRA**

**CREATE DATABASE jira\_db;**

**Next, create a user for JIRA**

**CREATE USER 'jira\_user'@'localhost' IDENTIFIED BY 'Your\_Password';**

**Grant all privileges to the JIRA user in order to access the JIRA database.**

**GRANT ALL PRIVILEGES ON jira\_db.\* TO 'jira\_user'@'localhost' IDENTIFIED by 'Your\_Password';**

**To wrap it up flush the privileges**

**FLUSH PRIVILEGES;**

## **Step 2: Download JIRA & assign necessary permissions**

**Having created the database for JIRA, our next step will be to download JIRA's binary file. Download the file using the** [**wget command**](https://linoxide.com/linux-command/cool-wget-examples/) **as shown.**

**# wget https://www.atlassian.com/software/jira/downloads/binary/atlassian-jira-software-7.8.2-x64.bin**

**Next, assign** [**execute permissions**](https://linoxide.com/linux-command/chmod-command/) **to the binary file**

**# chmod a+x atlassian-jira-software-7.8.2-x64.bin**

**WIth execute permissions assigned, run the binary file:**

**./ atlassian-jira-software-7.8.2-x64.bin**

**When running the binary file you will be prompted to answer a couple of questions as highlighted below.**

**Ensure you make the selections as highlighted below.**

**If the installation goes well as expected, you will get the output highlighted in. You can proceed and access JIRA on the web interface via port 8080. But first, let's open some ports on the** [**UFW firewall**](https://linoxide.com/linux-how-to/setup-ufw-ubuntu/) **to give access to external users to access JIRA.**

## **Step 3: Open JIRA ports**

**To successfully access and configure JIRA on the web console, we need to open a few ports on the UFW firewall that will enable us to access JIRA.**

**If UFW is not installed on your system, run the command**

**# apt install ufw**

**Now, you can proceed and open ports 8080, 8181 and 8005**

**# ufw allow 8181**

**# ufw allow 8005**

**# ufw allow 8080**

**Then enable the firewall for the rules to come into effect**

**# ufw enable**

**To confirm that the rules exist. run the command:**

**# ufw status**

## **Step 4: Accessing JIRA**

**All is now set to access JIRA and proceed with the web setup. Fire up your browser and browse your server's IP as shown:**

**http://server-IP:8080**

**The first page presents two options as shown. Click on 'I'll set it up myself' and click on the 'Next' button.**

**On the next Window, click on 'My own database' and for the Database type value, click and select 'MySQL'. You will get a notification that JIRA requires you to download a MySQL driver and thereafter restart JIRA after installing it.**

**To download the MySQL driver, use the wget command to download it from MySQL's official site.**

**wget https://dev.mysql.com/get/archives/mysql-connector-java-5.1/mysql-connector-java-5.1.36.zip**

**Once downloaded,** [**unzip the zipped file**](https://linoxide.com/linux-command/best-compression-with-zip-command/)

**# unzip mysql-connector-java-5.1.36.zip**

**Next, navigate into the mysql-connector-java directory and copy the jar file to the path /opt/atlassian/jira/lib**

**# cp mysql-connector-java-5.1.36-bin.jar /opt/atlassian/jira/lib**

**Next, restart JIRA using the commands:**

**# /etc/init.d/jira stop**

**# /etc/init.d/jira start**

**Now head back to your browser and fill out all the details and click on 'Test connection' button at the bottom.**

**StackStorm Jira Integration steps :**

# **JIRA integration pack**

**This pack consists of a sample JIRA sensor and a JIRA action.**

## **Installation**

**You will need to have gcc installed on your system. For Ubuntu systems, run sudo apt-get install gcc. For Redhat/CentOS systems, run sudo yum install gcc libffi-devel python-devel openssl-devel. To build the python cryptography dependency (part of the following st2 pack install command) 2GB of RAM is recommended. In some cases adding a swap file may eliminate strange gcc compiler errors.**

**Then install this pack with: st2 pack install jira**

## **Configuration**

**Copy the example configuration in** [**jira.yaml.example**](https://github.com/StackStorm-Exchange/stackstorm-jira/blob/master/jira.yaml.example) **to /opt/stackstorm/configs/jira.yaml and edit as required.**

* **url - URL of the JIRA instance (e.g. https://myproject.atlassian.net)**
* **poll\_interval - Polling interval - default 30s**
* **project - Key of the project which will be used as a default with some of the actions which don't require or allow you to specify a project (e.g. STORM).**
* **verify - Verify SSL certificates. Default True. Set to False to disable verification**
* **auth\_method - Specify either basic or oauth authentication**

**Include the following settings when using the oauth auth\_method:**

* **rsa\_cert\_file - Path to the file with a private key**
* **oauth\_token - OAuth token**
* **oauth\_secret - OAuth secret**
* **consumer\_key - Consumer key**

**Include the following settings when using the basic auth\_method:**

* **username - Username**
* **password - Password**

**If using the oauth auth\_method, take a look at the OAuth section below for further setup instructions.**

**You can also use dynamic values from the datastore. See the** [**docs**](https://docs.stackstorm.com/reference/pack_configs.html) **for more info.**

**Note : When modifying the configuration in /opt/stackstorm/configs/ please remember to tell StackStorm to load these new values by running st2ctl reload --register-configs**

### **OAuth**

### **Disclaimer**

**This documentation is written as of 06/17/2014. JIRA 6.3 implements OAuth1. Most of this doc would need to be revised when JIRA switches to OAuth2.**

### **Steps**

**Generate RSA public/private key pair  
# This will create a 2048 length RSA private key**

**$openssl genrsa -out mykey.pem 2048**

**# Now, create the public key associated with that private key**

**openssl rsa -in mykey.pem -pubout**

**Generate a consumer key. You can use python uuid.uuid4() to do this, for example:  
$ python**

**Python 2.7.10 (default, Jul 30 2016, 19:40:32)**

**[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.34)] on darwin**

**Type "help", "copyright", "credits" or "license" for more information.**

**>>> import uuid**

**>>> print uuid.uuid4()**

**210660f1-ca8a-40d5-a6ee-295ccbf3074d**

**>>>**

1. **Configure JIRA for external access:**
   * **Go to AppLinks section of your JIRA - https://JIRA\_SERVER/plugins/servlet/applinks/listApplicationLinks**
   * **Create a Generic Application with some fake URL**
   * **Click Edit, hit IncomingAuthentication. Plug in the consumer key and RSA public key you generated.**
2. **Get access token using this** [**script**](https://github.com/lakshmi-kannan/jira-oauth-access-token-generator/blob/master/generate_access_token.py)**. You may need to install additional libraries to run that script, and you will need to edit the script to use your file locations. Check the** [**README**](https://github.com/lakshmi-kannan/jira-oauth-access-token-generator/blob/master/README.md) **file for more information. The access token is printed at the end of running that script. Save these keys somewhere safe.**
3. **Plug in the access token and access secret into the sensor or action. You are good to make JIRA calls. Note: OAuth token expires. You'll have to repeat the process based on the expiry date.**

## **Sensors**

### **JiraSensor**

**The sensor monitors for new tickets and sends a trigger into the system whenever there is a new ticket.**

## **Actions**

* **create\_issue - Action which creates a new JIRA issue.**
* **get\_issue - Action which retrieves details for a particular issue.**

**Please follow this link for more detailed info**

[**https://github.com/StackStorm-Exchange/stackstorm-jira**](https://github.com/StackStorm-Exchange/stackstorm-jira)