StackStorm 中文手册 V2.7

—— 快速入门篇

https://docs.stackstorm.com/

2018-07

StackStorm 官方文档翻译群：438849333

StackStorm 中国用户群 QQ 群：138132636

AIOPStack.com AIOPStack.cn	JimChen 翻译, https://github.com/aiopstack/StackStormCN

目录

1 StackStorm 概述

https://docs.stackstorm.com/overview.html

1.1 关于 StackStorm

StackStorm 是跨服务和工具的集成和自动化平台。它结合现有的基础设施和应用程序环境，这样您可以更容易地自动化该环境。特别注重对事件响应中采用的动作。

StackStorm 帮助自动化常见的操作模式。一些应用例子如下：

- 便于故障排除—将会捕获到系统故障的各种触发，包括 Nagios、Sensu、New Relic 和其他监视系统，对物理节点、OpenStack 或 Amazon 实例以及应用程序组件进行一系列诊断检查，并将结果发布到如 HipChat 或 JIRA 这种共享通信上下文中。
- 自动修复 -识别和验证 OpenStack 计算节点上的硬件故障，正确地疏散实例，并向管理员发送电子邮件，说明潜在可能的停机时间，但如果有什么问题—将冻结工作流并调用 PagerTask 唤醒人工干预。
- 持续部署——与 Jenkins 一起进行持续构建和测试，提供一个新的 AWS 集群，使用负载均衡器开启一些流量，并根据 NewRelic 应用程序的性能数据进行自动地前滚或回滚。
- StackStorm 帮助您按照规则和工作流或操作编排这些操作模式和其他操作模式。这些规则和工作流以代码的方式存储在 StackStorm 平台中，这意味着它们支持与您现在用于代码开发时使用相同的协作方法。它们可以与更广泛的开源社区共享，例如通过 StackStorm 社区。

1.2 工作原理

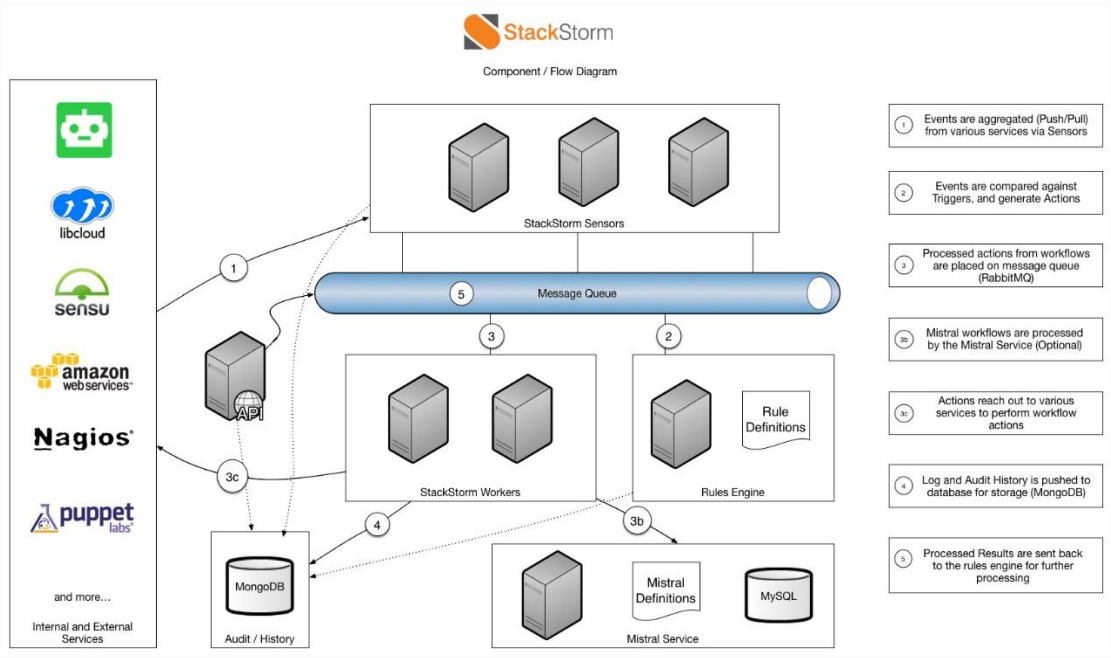


图 1-1　StackStorm 架构图

StackStorm 在环境中插入一系列可扩展适配器，包括 sensor（传感器）和 action 等。

- **Sensors**（传感器）是一些集成的 Python 插件，用于接收或监视事件的入站或出站。当来自外部系统的事件发生并由传感器处理时，将向 StackStorm 系统发出一个触发器。
- **Triggers**（触发器）是外部事件的 StackStorm 代理。有通用触发器(例如计时器、web 挂钩)和集成触发器(例如 Sensu 警报、JIRA 问题更新)。也可以通过编写传感器插件来定义新的触发器类型。
- **Actions**（动作）是 StackStorm 出站的集成部分。有通用动作(ssh、REST 调用)、集成动作(OpenStack、Docker、Puppet)或自定义动作。通过添加几行元数据，动作可以将 Python 插件或任何脚本集成到 StackStorm 中。Action 可以由用户通过 CLI 或 API 直接调用，也可以作为规则和工作流的一部分使用和调用。
- **Rules**（规则）将 trigger 映射到 action (或工作流)，按照匹配标准，把 trigger 有效负载映射到 action 的输入。
- **Workflows**（工作流）将动作整合到“uber-action”中，定义顺序、转换条件和传递数据。大多数自动化操作不止一步，因此需要多个操作。就像“原子”操作一样的工作流可以保存在 Action 库中，可以手动调用，也可

以由规则触发。

- Pack（包）是部署的内容单元。它们通过分组集成(触发器和操作)和自动化(规则和工作流)，简化了 StackStorm 可插拔内容的管理和分享。StackStorm Exchange 上有越来越多的包可用。用户可以创建自己的包，然后在 Gistub 上分享它们，也可以向 StackStorm Exchange 提交。
- **Audit trail**（审计跟踪）是指行动的手动或自动执行情况，触发器上下文和执行结果的全部细节都记录和存储下来。它也保存在审计日志中，可以是外部日志记录和分析工具集成：LogStash、Splunk、Statsd、SysLog 等。

StackStorm 是一个具有模块化架构的服务。它由松散耦合的服务组件组成，通过消息总线进行通信，并可以水平方式进行扩展，在不同规模上实现自动交付。StackStorm 既有 WebUI、CLI 客户端，当然也有完整的 REST API。还提供与 Python 绑定的客户端，便于开发人员使用。

StackStorm 是一种新型的、正在蓬勃发展的新产品。我们非常渴望与社会各界接触，听取您的反馈意见，并完善我们的产品方向。当然也非常欢迎捐款！

下一章节内容

- 按照 Installation 进行安装、运行
- 跟随 Quick Start 指导，构建一个简单的自动化
- 评价路线图（Roadmap），帮助我们指明发展方向
- 探索 StackStorm 社区

提问？问题？建议？都很欢迎！

- 支持论坛（Support Forum）
- Slack 社区频道：stackstorm-community.slack.com (注册 这里)
- 支持邮箱：support@stackstorm.com

2 安装 StackStorm

https://docs.stackstorm.com/install/index.html

准备好安装 StackStorm 了吗？以下是如何让系统启动和运行的概述。

StackStorm 是作为 RedHat/CentOS 和 Ubuntu Linux 系统的 RPM 和 Debs 以及 Docker 映像发布的。您可以使用脚本在单台系统上自动安装和配置所有组件，也可以选择您的操作系统按照手册说明进行安装。

以下是这些安装选项的简述：

- 单行安装（**One-line Install**）：运行我们的安装脚本，将在单台系统上安装所有组件。这是我们推荐的一种开始方式。有关详细信息，请参阅下面的 Quick Install 章节。
- 手动安装：是否有定制需求？也许你的服务器无法访问互联网？还是不喜欢用脚本安装？阅读您的操作系统相关的手动安装说明(Ubuntu 14/16, RHEL/CentOS 6, RHEL/CentOS 7)，并根据您的要求进行调整。下面是一些关于为 StackStorm Repos 设置内部镜像的附加指导（additional guidance ）。
- Ansible Playbook 安装：如果您是 Ansible 用户，请检查这些安装 StackStorm 的 Ansible Playbook。可重复、一致性、幂等性等是理想的 StackStorm 安装方式。
- **Docker 方式**安装：StackStorm 现在 Docker 上也得到了支持，请查看我们有关 Docker 的安装指南。
- **Vagrant 安装方式**：Vagrant 是一种快速提升测试系统的方法，参见 st2vagrant。这将创建一个新的 VM，并安装 StackStorm。

可以选择最适合您需求的安装选项。

升级到 Extreme Workflow Composer？这是一组安装在 StackStorm 之上额外的软件包。您可以一次安装 StackStorm 和 Extreme Workflow Composer，或者将 Extreme Workflow Composer 包添加到现有的 StackStorm 系统中。如果使用了 Extreme Workflow Composer，还可以添加 Network Automation Suites。有关更多信息，请参阅安装 Extreme Workflow Composer 文档。

1）快速安装

获取一个干净的符合系统要求的 64 位 Linux 系统。确保 `curl` 是最新的，然后在 Ubuntu 上使用 `sudo apt-get install curl` 或在 RHEL/CentOS 上用 `sudo yum install curl nss` 安装 curl。然后运行以下命令：

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

这是一个 StackStorm 安装的可选项。它将根据单台主机部署方式下载和安装所有组件。假设您已有一个干净的、基本的 Ubuntu 或 RHEL/CentOS 操作系统用于安装。

如果您尝试在运行其他应用程序或本地定制化环境的服务器上安装 StackStorm，则可能会遇到一些问题。在这种情况下，您应该使用某种手动安装方法。

脚本本身并不具有幂等性。如果尝试在失败的服务器上再重新运行脚本将失败。需要在一个干净的系统重新安装，或者切换到手动安装方式。

如果您需要通过代理方式进行安装，只需在运行脚本之前 export 出 `http_proxy`, `https_proxy`, `no_proxy` 环境变量。

```
export http_proxy=http://proxy.server.io:port
export https_proxy=http://proxy.server.io:port
export no_proxy=localhost,127.0.0.1
```

对于 MITM 代理的场景，您可能需要 export 出额外的 `proxy_ca_bundle_path`，请参见用代理方式安装包（Installing Packs from Behind a Proxy）。

如果访问在 RHEL 7/CentOS 7 系统上的 WebUI 有问题，请检查系统防火墙设置。

备注

出于安全考虑，安装程序脚本支持身份验证，并为相关服务(如 MongoDB 和 PostgreSQL)生成随机密码。

如果由于某种原因(例如，调试)，您需要直接访问这些服务，您可以在配置文件`/etc/st2/st2.conf`中找到 MongoDB 和 RabbitMQ 密码，或在`/etc/mistral/mistral.conf`中找到 PostgreSQL 密码。

2）其他安装选项

有关部署过程或特定于操作系统的安装说明的详细信息，请参阅下面内容：

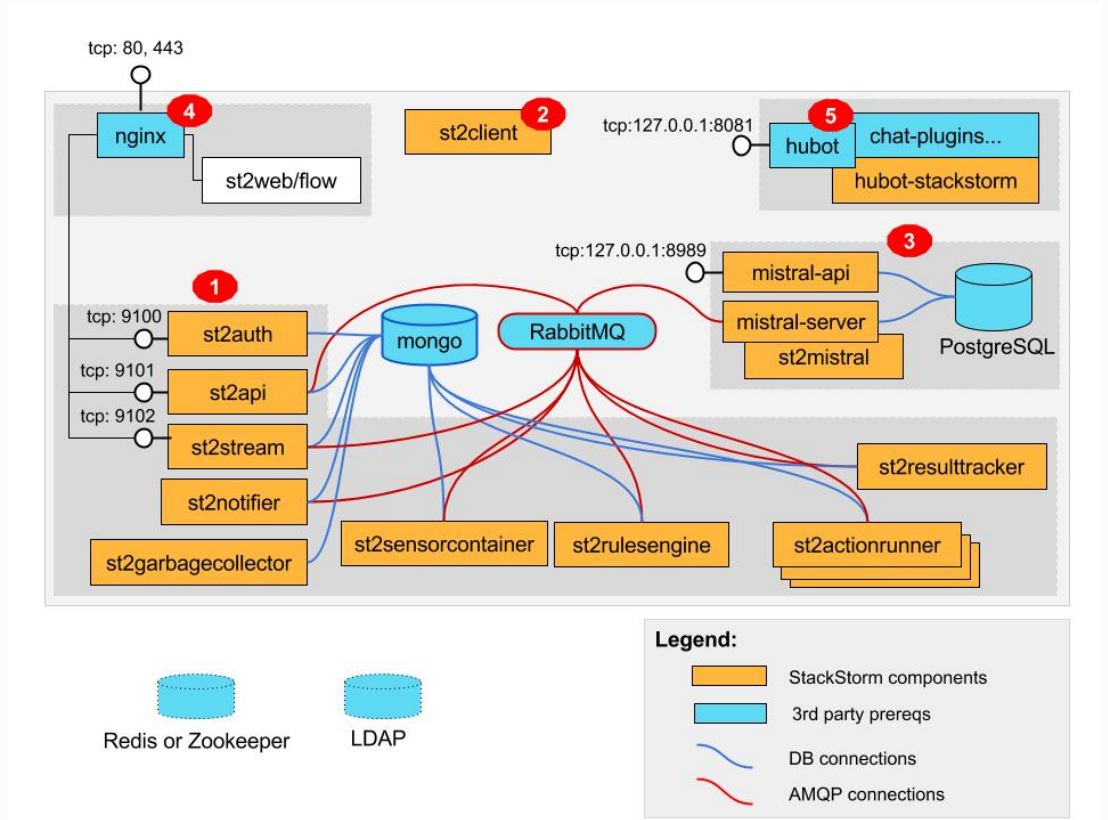
- 部署过程概述
- 安装系统要求
- Ubuntu 14.04 / 16.04
- RHEL 7 / CentOS 7
- RHEL 6 / CentOS 6
- Docker
- Ansible Playbooks
- Extreme Workflow Composer
- 配置 Configuration
- 升级 Upgrades
- 卸载 Uninstall

2.1 单台参考部署概述

https://docs.stackstorm.com/install/overview.html

本节描述了在单台系统上安装的所有 StackStorm 组件的“ 参考部署”。它解释了有哪些主要组件、它们的角色，以及怎么把它们串联在一起。

如果您使用一行快速安装脚本，将按照下图配置系统：



2.1.1. st2 服务

“st2”服务提供了 StackStorm 的主要功能。它们位于`/opt/stackstorm/st2`，共享一个通过`/etc/st2/st2.conf`进行配置专用的 Python 虚拟环境。

- **st2sensorcontainer** 运行`/opt/stackstorm/packs`目录下的传感器。它管理将在一个节点上运行的传感器，可以对它进行启动、停止和重启操作。
- **st2rulesengine** 在收到 TriggerInstance 时将对规则评判，并决定是否执行 ActionExecution 请求。它需要访问 MongoDB 来定位规则，而 RabbitMQ 侦听 TriggerInstance 和 ActionExecution 请求。此过程的另外用途是执行所有定义的定时器。
- **st2actionrunners** 通过各种 Action 执行器运行`/opt/stackstorm/packs`下的包含的 action 包。执行器可能需要一些针对特定执行器的配置，例如需要配置 SSH 才能运行基于`remote-shell-runner`和`remote-command-runner`的远程 action。而对于 Windows 需要先准备好 Windows 的执行器。详情见执行器（Runner）。
- **st2resultstracker** 通过调用 Mstral API 端点，跟踪长时间运行的工作

流的执行情况。

- **st2notifier** 在 ActionExecution 执行完成时 TriggerInstances 产生 `st2.core.actiontrigger` 和 `st2.core.notifytrigger`。另外一个目的是充当 action 的备份调度器，处理哪些可能未列入计划的 action。
- **st2garbagecollector** 是一个可选的服务，可以根据在 `/etc/st2/st2.conf` 中设置的参数，周期性地从数据库中删除那些老旧的执行历史数据。
- **st2auth** 是一个支持终端 REST 的认证服务。后端可用各种的认证方法，有关详细信息，请参阅身份验证（Authentication）。参考部署后端认证是使用平面文件（flat file）。
- **st2api** 是用于 CLI 和 Web UI 的 REST API Web 服务端点。它还为 webhook 提供 webhook 触发器。
- **st2stream** 是一个 HTTP 消费端的事件流，发布了各种有用的事件。这些事件由 Webui 和 Hubot 消费，如 Chatps 更新结果等。

2.1.2. st2client

`st2client` 是 CLI 和通过 StackStorm API 绑定的 Python。要将 CLI 配置成能指向正确的 API，请使用身份验证选项、禁止对自签名证书的不安全警告以及其他方便方式，请参阅 CLI 参考（CLI Reference）。`st2client` 与 `st2` 一起发行，也可以独立安装。

2.1.3. st2mistral

Mstral 是 StackStorm 为长时间执行的工作流使用的工作流服务组件。它被打包为 `st2mistral`，安装在 `/opt/stackstorm/mistral` 下，在一个专门的 Python 虚拟环境中运行，并通过 `/etc/mistral/mistral.conf` 进行配置。`mistral-server` 运行工作流逻辑和调用 action，向 st2api 提供 action 的执行请求。`st2mistral` 是一个随着 stackstorm 扩展的 mstral 插件。`mistral-api` 是由 `st2actionrunner` 和 `st2notifier` 访问的内部端点。在单台部署方式中，它只能限于 localhost 主机。

2.1.4. NGINX 的 WebUI 和 SSL 终结

- nginx 提供 SSL 终结，将 HTTP 重定向到 HTTPS，为 WebUI 静态组件服务，并为 RESTAPI 端点到 st2*的 Web 提供反向代理服务。
- StackStorm WebUI 包括 st2web、Workflow Designer、eXtreme Workflow Composer，安装在`/opt/stackstorm/static/webui`，通过`webui/config.js`配置。`st2web`内置了它自己的`deb`和`rpm`包。Workflow Designer 是作为`bwc-enterprise`软件包的一部分部署的，是 HTML 5 应用程序，充当静态 HTML，并调用 StackStorm 的 st2auth 和 st2api 的 RESTAPI 端点。Nginx 代理 st2api 和 st2auth 服务的`/api`和`/auth`的入站请求。

2.1.5. st2chatops – ChatOps 组件

StackStorm 的 Chatops 组件是 Hubot、st2 的 Hubot 适配器、连接到不同聊天服务的插件。它们打包为`st2chatops`，安装在`/opt/stackstorm/chatops/`，在`/opt/stackstorm/chatops/st2chatops.env`文件中配置。

还可以通过在已有 Hubot 实例上安装 hubot-stackstorm plugin 插件来启用 ChatOps。

2.1.6. 依赖软件

必须的依赖软件包括 RabbitMQ、MongoDB、PostgreSQL，可选的依赖软件如下：

- nginx 用于 SSL 终结、反向代理 API 端点和服务
- 并发策略（Policies）使用的 Redis 或者 Zookeeper
- Extreme Workflow Composer 中 LDAP 认证的 LDAP

2.1.7. 多主机环境或高可用（HA）部署

有关实现 HA 或水平扩展的多主机部署的具体信息，请参阅高可用性部署（High Availability Deployment）。

2.2 系统要求

https://docs.stackstorm.com/install/system_requirements.html

StackStorm 需要 Ubuntu、RHEL 或 CentOS 的 Linux 环境。不支持其他任何 Linux 发行版。下表列出了支持的 Linux 版本，以及我们用于测试的 Vagrant Box 和 Amazon AWS 实例。

如果您是从 ISO 镜像安装，只要执行最小的安装。对于 Ubuntu，可以使用 Server 版的变体，只需要基本集合中再添加 OpenSSH 服务包。在安装 StackStorm 时，所有其他依赖软件包都将自动添加。

备注

请注意，StackStorm 只支持 64 位架构。

Linux (64-bit)	Vagrant Box	Amazon AWS AMI
Ubuntu 14.04	bento/ubuntu-14.04	Ubuntu Server 14.04 LTS (HVM)
Ubuntu 16.04	bento/ubuntu-16.04	Ubuntu 16.04 LTS - Xenial (HVM)
RHEL 7 / CentOS 7	bento/centos-7.2	Red Hat Enterprise Linux (RHEL) 7.2 (HVM)
RHEL 6 / CentOS 6	bento/centos-6.7	Red Hat Enterprise Linux (RHEL) 6 (HVM)

或者，您可以使用 StackStorm 的 Docker 镜像安装。您至少需要 1.13.0 版本的 Docker 引擎，以及 `docker-compose` 安装可选项。更多信息请参阅我们的 Docker 指南。

下面是测试环境和生产环境部署 StackStorm 的推荐最小配置：

Testing 测试环境	Production 生产环境
• Dual CPU • 2GB RAM • 10GB storage • Recommended EC2: **t2.medium**	• Quad core CPU • >16GB RAM • 40GB storage • Recommended EC2: **m4.xlarge**

备注

If you are planning to add the DC Fabric Automation Suite to your system later, you will need additional RAM. Check the DC Fabric Automation Suite System Requirements

如果您计划稍后向您的系统添加 DC Fabric Automation Suite，您将需要增加 RAM。请检查 DC Fabric Automation Suite 系统要求。

如果将文件系统分成多个分区和挂载点，请确保在`/var`和`/opt`中至少有 1GB 的可用空间。如果没有足够的可用空间，RabbitMQ 和 MongoDB 可能无法正确运行。

默认情况下，StackStorm 和相关服务将使用下面这些 TCP 端口：

- nginx (80, 443)
- mongodb (27017)
- rabbitmq (4369, 5672, 25672)
- postgresql (5432)
- st2auth (9100)
- st2api (9101)
- st2stream (9102)

如果系统中其他服务已经使用了这些端口，StackStorm 可能无法成功安装或正确运行。

2.3 Ubuntu Trusty/Xenial 环境安装

https://docs.stackstorm.com/install/deb.html

如果你只是在寻找一个快速的“单行”安装，检查顶层安装指南。如果您需要定制安装，请使用本指南一步一步的说明，将按部署指南在单个系统上安装 StackStorm。

备注

Luke，最好使用源码！我们努力保持文档及时更新，但是要真正了解发生的事情，最好的方法是查看安装脚本的代码。

2.3.1 系统要求

请检查支持版本和系统要求。

2.3.2 最小安装

1）安装依赖包

> **备注**
>
> 目前首选和受支持的 MongoDB 版本是 3.4。这也是安装脚本中安装的版本。StackStorm v2.2.0 和更高版本都支持 MongoDB 3.4。较旧的 StackStorm 版本(在 v1.6.0 之前)只支持 MongoDB 2.x。

安装 MongoDB、RabbitMQ 和 PostgreSQL：

```
sudo apt-get update
sudo apt-get install -y gnupg-curl
sudo apt-get install -y curl
```

```
# Add key and repo for the latest stable MongoDB (3.4)
wget -qO - https://www.mongodb.org/static/pgp/server-3.4.asc | sudo apt-key add -
sudo sh -c "cat <<EOT > /etc/apt/sources.list.d/mongodb-org-3.4.list
deb http://repo.mongodb.org/apt/ubuntu $(lsb_release -c | awk '{print $2}')/mongodb-org/3.4
multiverse
EOT"
sudo apt-get update

sudo apt-get install -y mongodb-org
sudo apt-get install -y rabbitmq-server
sudo apt-get install -y postgresql
```

对于 `Xenial` 版本的 Ubuntu，你可能需要使能 MongoDB 服务并启动

```
sudo systemctl enable mongod
sudo systemctl start mongod
```

2）创建软件仓库

下面的脚本将检测您的平台和架构，并设置合适的 StackStorm 软件仓库。它还会添加用于软件包签名使用的 GPG 密钥。

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

3）安装 StackStorm 组件

```
sudo apt-get install -y st2 st2mistral
```

如果您没在同一个系统上运行 RabbitMQ、MongoDB 或 PostgreSQL，或者需要修改默认参数，请到下列设置修改：

- RabbitMQ 连接到 `/etc/st2/st2.conf` 和 `/etc/mistral/mistral.conf`
- MongoDB 在 `/etc/st2/st2.conf`
- PostgreSQL 在 `/etc/mistral/mistral.conf`

详见配置文档章节

4）创建加密的数据存储

键值对存储允许用户存储加密的值(秘密)。这些存储的数据使用了 AES256 对称加密算法。要生成密码密钥，请执行下面指令：

```
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
```

5）创建 Mistral 数据库

执行下面指令在 PostgreSQL 中创建 Mistral 数据库：

```
# Create Mistral DB in PostgreSQL
cat << EHD | sudo -u postgres psql
CREATE ROLE mistral WITH CREATEDB LOGIN ENCRYPTED PASSWORD 'StackStorm';
CREATE DATABASE mistral OWNER mistral;
EHD

# Setup Mistral DB tables, etc.
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf upgrade head
# Register mistral actions
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf populate |
grep -v -e openstack -e keystone
```

6）配置 SSH 和 SUDO

要运行本地和远程的操作，StackStorm 使用特殊的系统用户(默认情况下为 `stanley`)。对于远程 Linux 操作将使用 SSH。建议在所有远程主机上配置基于公钥的 SSH 访问。也建议将 SSH 访问配置到本地主机，便于运行示例和进行测试。

- 创建 StackStorm 系统用户，启用无密码 sudo，并设置 ssh 可以访问“localhost”，便于在本地测试基于 SSH 的操作。做这些配置您需要提升用户特权；

```
# 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
```

- 配置 SSH 访问，并在 StackStorm 主机能够通过 SSH 执行远程操作，远程主机上启用无密码 sudo。使用上一步中生成的公钥，并按照配置 SSH 中的说明操作。若要控制 Windows 系统，需要配置访问 Windows 执行器。
- 如果使用不同的用户或不同的 SSH 密钥路径，需要在 `/etc/st2/st2.conf` 中对应的配置项进行修改；

```
[system_user]
user = stanley
ssh_key_file = /home/stanley/.ssh/stanley_rsa
```

7）启动服务

- 启动服务：

```
sudo st2ctl start
```

- 注册 sensor、rule 和 action

```
sudo st2ctl reload
```

8）验证

用以下命令进行测试 StackStorm 的安装，这些都应顺利的完成：

```
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
```

使用 st2ctl 脚本管理 StackStorm 服务：

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

此时，已经可以使用最小安装可工作的系统了，并且可以愉快地使用 StackStorm：按照快速入门教程、部署实例，并探索如何从 StackStorm Exchange 中安装 pack。

但是如果没有 WebUI 就会缺少欢乐，没有 SSL 或身份验证就没法保证安全，没有 ChatOps 就会没有乐趣，没有钱就没有 Extreme Workflow Composer。请继续往下读！

2.3.3 配置身份认证

为了简单起见，参考的部署使用基于文件的身份验证程序。请参考身份验证来配置和使用 PAM 或 LDAP 的后端身份验证。

创建基于文件的身份认证：

- 创建用户口令

```
# 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
```

- 在`/etc/st2/st2.conf`文件中启用和配置身份认证：

```
[auth]
# ...
enable = True
backend = flat_file
backend_kwargs = {"file_path": "/etc/st2/htpasswd"}
```

```
# ...
```

- 重启 st2api 服务：

```
sudo st2ctl restart-component st2api
```

- 验证用户、设置环境变量、检查运行情况：

```
# Get an auth token to use in CLI or API
st2 auth st2admin

# A shortcut to authenticate and export the token
export ST2_AUTH_TOKEN=$(st2 auth st2admin -p 'Ch@ngeMe' -t)

# Check that it works
st2 action list
```

查看 CLI 参考指南可以学到其他通过 CLI 方式配置用户验证。

2.3.4 安装 WebUI 和创建 SSL 终结器

Nginx 用于服务 WebUI 静态文件、将 HTTP 重定向到 HTTPS、提供 SSL 终端以及反向代理 st2auth 和 st2api API 端点。创建 Nginx 需要安装 `st2web` 和 `nginx` 软件包，生成证书或将现有证书放在 `/etc/ssl/st2` 下，并使用 StackStorm 提供的站点配置文件 st2.conf 配置 nginx。

StackStorm 依赖于 Nginx 版本>=1.7.5。在 Ubuntu 标准的软件包仓库中有一个旧版本，因此需要添加 Nginx 官方的软件仓库：

```
# Add key and repo for the latest stable nginx
sudo apt-key adv --fetch-keys http://nginx.org/keys/nginx_signing.key
sudo sh -c "cat <<EOT > /etc/apt/sources.list.d/nginx.list
deb http://nginx.org/packages/ubuntu/ $(lsb_release -c | awk '{print $2}') nginx
EOT"
sudo apt-get update

# Install st2web and nginx
# note nginx should be > 1.4.6
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
```

如果在 nginx 配置中修改端口、url 路径，在 st2web 中的配置文件 `/opt/stackstorm/static/webui/config.js` 也要对应修改。

使用浏览器访问 `https://${ST2_HOSTNAME}`，登录 WebUI。

如果从主机外部访问 API，已经按照下面说明配置了 nginx，使用 `https://${EXTERNAL_IP}/api/v1/${REST_ENDPOINT}` 访问。

例如：

```
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
```

类似的，可以使用 `https://${EXTERNAL_IP}/auth/v1/${AUTH_ENDPOINT}` 与认证的 REST 端点连接。

可以在命令行中添加 `--debug` 选项，查看资源的实际 REST 端点。

例如，查看获取端点的 action、invoke：

```
st2 --debug action list
```

2.3.5 创建 ChatOps

如果已经运行了一个 Hubot 实例，可以安装 hubot-stackstorm 插件并配置如下所述的 StackStorm 环境变量。其他，启用 StackStorm ChatOps 最简单的方法是使用 st2chatops 软件包。

Validate that the `chatops` pack is installed, and a notification rule is enabled:

验证 `chatops` 软件包 pack 已经安装，并启用一个通知的 rule：

```
# Ensure chatops pack is in place
ls /opt/stackstorm/packs/chatops
# Create notification rule if not yet enabled
st2 rule get chatops.notify || st2 rule create /opt/stackstorm/packs/chatops/rules/notify_hubot.yaml
```

添加 NodeJS v6 软件仓库：

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
```

安装 `st2chatops` 软件包：

```
sudo apt-get install -y st2chatops
```

检查并编辑`/opt/stackstorm/chatops/st2chatops.env`配置文件，找到正在使用的 StackStorm 中安装和使用的聊天服务。至少需要生成一个 API 密钥并设置 `ST2_API_KEY` 变量。默认情况下，`st2api` 和 `st2auth` 应该位于同一个主机上。如果不是这样，请更新 `ST2_API` 和 `ST2_Auth_URL` 变量，或者用 `ST2_HOSTNAME` 指向正确的主机。

示例配置中使用了 Sack。设置时需要挑转到 Slack 的 web 管理界面，创建一个 Bot，然后将身份验证令牌复制到 `HUBOT_SLACK_TOKEN` 中。

如果您使用的是不同的聊天服务，请在 `st2chatops.env` 中的 `Chat service adapter settings` 部分下设置相应的环境变量：Slack、HipChat、Yammer、Flowdock、IRC 、XMPP。

启动服务：

```
sudo service st2chatops start
```

重新加载 st2 pack，确保注册的 `chatops.notify` rule 生效：

```
sudo st2ctl reload --register-all
```

就这样!进入到你的聊天室开始 ChatOps。在 ChatOps 章节阅读更多内容。

2.3.6 安全注意事项

默认情况下，在安装 MongoDB、RabbitMQ 和 PostgreSQL 时，它们禁用身份验证或使用默认静态密码。因此在安装完这些服务之后，您应该修改配置，使用强随机生成的密码启用身份验证。

注意：如果您使用 StackStorm 安装脚本，这些将自动为您完成。

为这些服务配置授权和密码超出了本文档的范围。有关更多信息，请参阅以下链接：

- MongoDB：https://docs.mongodb.com/manual/tutorial/enable-authentication/, https://docs.mongodb.com/manual/core/authorization/
- RabbitMQ：https://www.rabbitmq.com/authentication.html
- PostgreSQL：https://www.postgresql.org/docs/9.4/static/auth-methods.html

启用这些组件的身份验证之后，需要重载 StackStorm 服务使用新设置。

也就是编辑下列配置选项：

1）StackStorm - `/etc/st2/st2.conf`

- `database.username` – MongoDB 数据库用户名
- `database.password` – MongoDB 数据库口令
- `messaging.url` – RabbitMQ 传输 url (`amqp://<username>:<password>@<hostname>:5672`)

2）Mistral - `/etc/mistral/mistral.conf`

- `database.connection` – PostgreSQL 数据库连接字符串 (`postgresql+psycopg2://<username>:<password>@<hostname>/mistral`)
- `transport_url` – RabbitMQ 传输 url (`rabbit://<username>:<password>@<hostname>:5672`)

此外，强烈鼓励您遵循以下运行网络服务的最佳实践：

- 确保服务之间的通信是加密的。为 MongoDB、RabbitMQ 和 PostgreSQL 启用 SSL/TLS。
- 将服务配置为只侦听本地主机，并在需要时侦听内部 IP 地址。通常大多数服务不需要使用，但 StackStorm 的 MongoDB、RabbitMQ、PostgreSQL 运行在有公共 IP 地址上是可用。
- 配置防火墙并创建白名单。防火墙应该只允许那些需要访问这些服务的用户和系统访问。API 和 auth 服务通常需要提供给用户访问，但其他的依赖服务，如 MongoDB、RabbitMQ 和 PostgreSQL 则不需要访问。这些组件不用让用户直接访问，而只允许 StackStorm 组件与其直接通信。
- 在可能的情况下，还应该利用其他基于网络的隔离和安全特性，如 DMZ。

上面提到的步骤对于分布在多个服务器上运行 StackStorm 组件的生产部署方式特别重要。

2.3.7 升级 Extreme Workflow Composer 软件

Extreme Workflow Composer 将工作流设计器(用于工作流创建/编辑的图形工具)、RBAC 和 LDAP 添加到 StackStorm 中。它作为一组附加的软件包部署在 StackStorm 之上。您将需要一个激活的 Extreme Workflow Composer 订阅和一个许可密钥来访问 Extreme Workflow Composer 软件仓库。

学习更多关于 Extreme Workflow Composer 内容，咨询报价、申请评估版授权，请访问网站 stackstorm.com/product

要安装 Extreme Workflow Composer，请将下面命令中的 `${EWC_License_Key}` 替换成注册或购买时收到的密钥，并运行以下命令：

```
# Set up Extreme Workflow Composer repository access
curl -s
https://${EWC_LICENSE_KEY}:@packagecloud.io/install/repositories/StackStorm/enterprise/script.deb.sh | sudo bash
# Install Extreme Workflow Composer
sudo apt-get install -y bwc-enterprise
```

下一章节内容？

- ❑ 按照快速入门指南创建一个简单的自动化
- ❑ 获取更多的 action、trigger 和 rule
  - ■ 按照 Packs 指南从 StackStorm Exchange 安装集成 pack；
  - ■ 把脚本转成 StackStorm 的 actions；
  - ■ 学习如何编写自己的 action
- ❑ Workflows—使用工作流把 action 串成更高层次的自动化；
- ❑ 查看 stackstorm.com 上的教材；

2.4 RHEL 7/CentOS 7 环境安装

https://docs.stackstorm.com/install/rhel7.html

如果您只是在寻找一个快速的“一键”安装，检查顶层安装指南。如果您需要定制化安装，请根据参考部署使用指南一步一步地在单台系统上安装 StackStorm。

备注

卢克，请使用源代码！我们努力保持文档是最新版本，但是要了解真正发生的事情最好的方法是查看安装脚本的代码。

2.4.1 系统安装要求

请参考支持软件版本和系统要求。

2.4.2 最小安装

1）调整 SELinux 策略

如果您的系统使用 SELinux 的 Enforcing 模式，请按照下面的说明调整 SELinux 策略。这是成功安装所必需的。如果您不想调整这些策略，您也可以根据您的安全实践对它们进行调整。

首先，请检查 SELinux 是否使用 Enforcing 模式：

```
getenforce
```

如果上面命令返回是'Enforcing'，那么执行下面命令：

```
# SELINUX management tools, not available for some minimal installations
sudo yum install -y policycoreutils-python
```

```
# Allow network access for nginx
sudo setsebool -P httpd_can_network_connect 1

# Allow RabbitMQ to use port '25672', otherwise it will fail to start
sudo semanage port --list | grep -q 25672 || sudo semanage port -a -t amqp_port_t -p tcp 25672
```

备注

如果您看到诸如“SELinux: Could not downgrade policy file”之类的消息，这意味着 SELinux 是 disabled 模式，您正在尝试调整策略配置。您可以忽略这个错误。

2）安装依赖软件

备注

目前的首选和受支持的 MongoDB 是 3.4 版本。这也是安装脚本安装的版本。StackStorm v2.2.0 和更高版本支持 MongoDB3.4。StackStorm 的旧版本(在 V1.6.0 之前)只支持 MongoDB2.x。

安装 MongoDB、RabbitMQ、PostgreSQL 数据库：

```
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm

# Add key and repo for the latest stable MongoDB (3.4)
sudo rpm --import https://www.mongodb.org/static/pgp/server-3.4.asc
sudo sh -c "cat <<EOT > /etc/yum.repos.d/mongodb-org-3.4.repo
[mongodb-org-3.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc
EOT"

sudo yum -y install mongodb-org
sudo yum -y install rabbitmq-server
sudo systemctl start mongod rabbitmq-server
sudo systemctl enable mongod rabbitmq-server
```

```
# Install and configure postgres
sudo yum -y install postgresql-server postgresql-contrib postgresql-devel

# Initialize PostgreSQL
sudo postgresql-setup initdb

# Make localhost connections to use an MD5-encrypted password for authentication
sudo sed -i "s/\(host.*all.*all.*127.0.0.1\/32.*\)ident/\1md5/" /var/lib/pgsql/data/pg_hba.conf
sudo sed -i "s/\(host.*all.*all.*::1\/128.*\)ident/\1md5/" /var/lib/pgsql/data/pg_hba.conf

# Start PostgreSQL service
sudo systemctl start postgresql
sudo systemctl enable postgresql
```

3）创建软件仓库

下面的脚本将检测您的平台和体系结构，并设置合适的 StackStorm 软件仓库。还将添加用于包签名的 GPG 密钥。

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

4）安装 StackStorm 软件组件

```
sudo yum install -y st2 st2mistral
```

如果您没有在同一个系统上运行 RabbitMQ、MongoDB 或 PostgreSQL，或者更改了默认值，请调整以下参数设置：

- RabbitMQ 连接配置在`/etc/st2/st2.conf`和`/etc/mistral/mistral.conf`
- MongoDB 配置在`/etc/st2/st2.conf`
- PostgreSQL 配置在`/etc/mistral/mistral.conf`

详见配置文档章节。

5）Datastore 加密

键值对存储允许用户存储加密的值(秘密)。这些存储使用对称加密算法(AES256)。要生成密码密钥，请运行以下命令：

```
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
```

6）创建 Mistral 数据库

运行以下这些命令来创建 Mstral PostgreSQL 数据库：

```
# Create Mistral DB in PostgreSQL
cat << EHD | sudo -u postgres psql
CREATE ROLE mistral WITH CREATEDB LOGIN ENCRYPTED PASSWORD 'StackStorm';
CREATE DATABASE mistral OWNER mistral;
EHD

# Setup Mistral DB tables, etc.
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf upgrade head
# Register mistral actions
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf populate |
grep -v -e openstack -e keystone
```

7）配置 SSH 和 SUDO

要运行本地和远程 shell 的 action 操作，StackStorm 使用特定的系统用户(默认情况下为 `Stanley`)。对于远程 Linux 使用 SSH 进行操作。建议在所有远程主机上配置基于公钥的 SSH 访问。我们也建议将访问本地主机也配置成 SSH 访问，以便运行示例和进行测试。

- 创建 StackStorm 系统用户，设置成无密码的 Sudo，并将“本地主机”设置为 SSH 访问，以便于在本地测试基于 SSH 的操作。有些您可能也需要提升权限才能执行操作：

```
# 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
```

- 在远程主机上配置 SSH 访问并启用无密码的 sudo，StackStorm 将通过 SSH 执行远程的 action 操作。使用上一步生成的公钥，按照配置 SSH 中的说明进行操作。如要控制 Windows 服务器，需要配置 Windows 执行器的访问权限。
- 如果您正在使用其他用户或带路径的 SSH 密钥，则需要在 `/etc/st2/st2.conf` 中更改此部分：

```
[system_user]
user = stanley
ssh_key_file = /home/stanley/.ssh/stanley_rsa
```

8）启动服务

- 启动服务:

```
sudo st2ctl start
```

- 注册传感器、规则和 action 操作：

```
sudo st2ctl reload
```

9）安装验证

以下命令将测试您安装的 StackStorm。它们都应该成功地执行完成：

```
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
```

使用脚本管理 StackStorm 服务：

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

在这一点上，您有一个很小的安装工作，并可以愉快地玩 StackStorm：请按照快速入门教程、部署示例、探索和从 StackStorm Exchange 安装 pack 软件包。

但是还不能享受 WebUI 的乐趣，安全上也没有 SSL 或身份验证，也没有 ChatOps 的欢乐，没有钱也就没有 Extreme Workflow Composer。请继续往下读！

2.4.3 配置身份认证

为了简单起见，使用基于文件的身份验证的参考部署。请参考身份验证来配置，并使用 PAM 或 LDAP 后端的身份验证。

创建基于文件的身份认证：

- 创建用户、设置口令：

```
# Install htpasswd utility if you don't have it
sudo yum -y install httpd-tools
# Create a user record in a password file.
echo 'Ch@ngeMe' | sudo htpasswd -i /etc/st2/htpasswd st2admin
```

- 在中`/etc/st2/st2.conf`启用并配置身份认证：

```
[auth]
# ...
enable = True
backend = flat_file
backend_kwargs = {"file_path": "/etc/st2/htpasswd"}
# ...
```

- 重启 st2api 服务：

```
sudo st2ctl restart-component st2api
```

- 身份验证、设置环境变量的令牌，并检查它是否有效：

```
# Get an auth token to use in CLI or API
st2 auth st2admin

# A shortcut to authenticate and export the token
export ST2_AUTH_TOKEN=$(st2 auth st2admin -p 'Ch@ngeMe' -t)

# Check that it works
st2 action list
```

查看 CLI 参考指南，了解其他通过 CLI 进行身份验证的简便方法。

2.4.4 安装 WebUI，创建 SSL 终端

Nginx 用于服务 WebUI 静态文件、将 HTTP 重定向到 HTTPS、提供 SSL 终端以及反向代理 st2auth 和 st2api 的 API 端点。要设置它：需要安装 `st2web` 和 `nginx` 软件包，生成证书或将已有证书放在 `/etc/ssl/st2` 下，然后配置 StackStorm 站点的 nginx 服务的配置文件 st2.conf。

StackStorm 依赖于版本>=1.7.5 的 Nginx。在 RHEL 软件仓库中有一个旧版本，因此您需要添加官方的 Nginx 软件仓库：

```
# Add key and repo for the latest stable nginx
sudo rpm --import http://nginx.org/keys/nginx_signing.key
sudo sh -c "cat <<EOT > /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/rhel/$releasever/x86_64/
gpgcheck=1
enabled=1
EOT"

# Install nginx, avoid epel repo in favor of nginx.org
sudo yum --disablerepo='epel' install -y nginx

# Install st2web
sudo yum install -y st2web

# Generate a 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 365 -nodes -subj "/C=US/ST=California/L=Palo Alto/O=StackStorm/OU=Information \
Technology/CN=$(hostname)"
```

```
# Copy and enable the supplied nginx config file
sudo cp /usr/share/doc/st2/conf/nginx/st2.conf /etc/nginx/conf.d/

# Disable default_server configuration in existing /etc/nginx/nginx.conf
sudo sed -i 's/default_server//g' /etc/nginx/nginx.conf

sudo systemctl restart nginx
sudo systemctl enable nginx
```

如果您需要修改 nginx 配置中的端口或 url 路径，那么需要在配置文件 `/opt/stackstorm/static/webui/config.js` 上对 st2web 配置选项进行相应的更改。

用您的浏览器访问 `https://${ST2_HOSTNAME}`，登录 WebUI。

Web 浏览器如果无法访问到，则可能需要更改默认防火墙设置。您可以使用以下命令来完成操作：

```
firewall-cmd --zone=public --add-service=http --add-service=https
firewall-cmd --zone=public --permanent --add-service=http --add-service=https
```

这将允许入站 HTTP(端口 80)和 HTTPS(端口 443)通信，并使这些更改在重新启动后还能生效。

如果您正在尝试从服务器外部访问 API，并且按照以下说明配置了 nginx，请使用 `https://${EXTERNAL_IP}/api/v1/${REST_ENDPOINT}`。

例如：

```
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
```

类似地，您可以使用 `https://${EXTERNAL_IP}/auth/v1/${AUTH_ENDPOINT}` 连接到认证的 REST 端点。

通过向相应资源的 CLI 命令添加 `--debug` 调试选项，可以看到资源的实际 REST 端点。

例如，要查看获取端点的 action 操作，请调用：

```
st2 --debug action list
```

2.4.5 创建 ChatOps

如果您已经运行了 Hubot 实例，您可以如下所述方式安装 hubot-stackstorm plugin 插件并配置 StackStorm 的环境变量。否则，启用 StackStorm ChatOps 最简单的方法是使用 st2chatops 软件包。

- 验证是否安装了 `chatops` 软件包，并启用通知规则：

```
# Ensure chatops pack is in place
ls /opt/stackstorm/packs/chatops
# Create notification rule if not yet enabled
st2 rule get chatops.notify || st2 rule create /opt/stackstorm/packs/chatops/rules/notify_hubot.yaml
```

- 添加 NodeJS v6 软件仓库

```
curl -sL https://rpm.nodesource.com/setup_6.x | sudo -E bash -
```

- 安装 `st2chatops` 软件包

```
sudo yum install -y st2chatops
```

- 检查和编辑 `/opt/stackstorm/chatops/st2chatops.env` 配置文件，将其指向您正在使用的 StackStorm 安装和聊天服务。至少，您应该生成一个 API 密钥并设置 `ST2_API_KEY` 变量。默认情况下，`st2api` 和 `st2auth` 应该位于同一个主机上。如果不是这样，请更新 `ST2_API` 和 `ST2_Auth_URL` 变量，或者用 `ST2_HOSTNAME` 指向正确的主机。

使用了 Slack 示例配置。若要创建这个配置，请转到 Slack 的 Web 管理页面，创建一个 BOT、并将身份验证令牌复制到 `HUBOT_SLACK_TOKEN` 中。

如果您使用的是不同的聊天服务，请在 `st2chatops.env` 中：Slack、HipChat、Yammer、Flowdock、IRC 、XMPP 中的 `Chat service adapter settings` 部分下设置相应的环境变量。

启动服务：

```
sudo systemctl start st2chatops

# Start st2chatops on boot
sudo systemctl enable st2chatops
```

重新装入 st2 的 pack 软件包，确保 `chatops.notify` 规则已经注册：

```
sudo st2ctl reload --register-all
```

就这样，到你的聊天室开始聊天。在 ChatOps 部分阅读更多内容。

2.4.6 安全注意事项

默认情况下，在安装了 MongoDB、RabbitMQ 和 PostgreSQL 后，它们禁用身份验证或使用默认静态密码。因此，在安装完成这些服务之后，您应该调整配置，使用增强随机生成的密码作为身份验证。

注意：如果您使用 StackStorm 安装脚本，将会自动为您完成。

为这些服务配置授权和密码超出了本文档的范围。有关更多信息，请参阅以

下链接：

- MongoDB - https://docs.mongodb.com/manual/tutorial/enable-authentication/, https://docs.mongodb.com/manual/core/authorization/
- RabbitMQ - https://www.rabbitmq.com/authentication.html
- PostgreSQL - https://www.postgresql.org/docs/9.4/static/auth-methods.html

启用这些组件的身份验证之后，还需要重启 StackStorm 服务，使新设置生效。

这意味着修改下面配置选项：

1）StackStorm - `/etc/st2/st2.conf`

- `database.username` - MongoDB database username.
- `database.username` – MongoDB 数据库用户名
- `database.password` – MongoDB 数据库口令
- `messaging.url` – RabbitMQ 传输 url(`amqp://<username>:<password>@<hostname>:5672`)

2）Mistral - `/etc/mistral/mistral.conf`

- `database.connection` – PostgreSQL 数据库连接字符串 (`postgresql+psycopg2://<username>:<password>@<hostname>/mistral`)
- `transport_url` – RabbitMQ 传输 url (`rabbit://<username>:<password>@<hostname>:5672`)

此外，强烈鼓励您按照以下这些运行网络服务的最佳实践：

- 确保服务之间的通信是加密的。为 MongoDB、RabbitMQ 和 PostgreSQL 之间启用 SSL/TLS。
- 将服务配置为只侦听本地主机，并在需要时侦听内部 IP 地址。通常 StackStorm(MongoDB、RabbitMQ、PostgreSQL) 大多数服务不需要使用在公共上可用 IP 地址。
- 配置防火墙并设置白名单。防火墙应该只允许那些需要访问这些服务的用户和系统可以访问。用户通常需要对 API 和 auth 服务进行访问，但其

他依赖服务，如 MongoDB、RabbitMQ 和 PostgreSQL 则不需要访问。用户不应直接访问这些组件，而只允许 StackStorm 组件与他们通信。

- 在可能的情况下，您还应该利用其他基于网络的隔离和安全特性，如 DMZ。

上面提到的步骤对于在多个服务器上运行 StackStorm 组件的分布式生产部署特别重要。

2.4.7 升级到 Extreme Workflow Composer

Extreme Workflow Composer 将工作流设计器(用于工作流创建/编辑的图形工具)、RBAC 和 LDAP 添加到 StackStorm 中。它作为一组附加的软件包部署在 StackStorm 之上。您将需要一个 Extreme Workflow Composer 的激活的订阅和一个许可证密钥来访问 Extreme Workflow Composer 的存储仓库。

要了解有关 Extreme Workflow Composer 的更多信息，查询报价或获得评估许可，请访问 stackstorm.com/product。

要安装 Extreme Workflow Composer，请将下面命令中的`${EWC_LICENSE_KEY}`替换为注册或购买时收到的密钥，并执行以下命令：

```
# Set up Extreme Workflow Composer repository access
curl -s
https://${EWC_LICENSE_KEY}:@packagecloud.io/install/repositories/StackStorm/enterprise/script.rpm.sh | sudo bash
# Install Extreme Workflow Composer
sudo yum install -y bwc-enterprise
```

下一章节内容

- 查看快速入门指南创建简单的自动化。
- 了解更多的 action、trigger、rule
  - ✓ 按照 Packs 指南，从 StackStorm Exchange 安装集成的 pack 软件包
  - ✓ 把您的脚本转成 StackStorm 的 action；
  - ✓ 学习如何编写自定义 action；
- 使用工作流将 action 整合到更高级别的自动化中—工作流 Workflows
- 查看 stackstorm.com 上的教程

2.5 RHEL 6/CentOS 6 环境安装

https://docs.stackstorm.com/install/rhel6.html

如果您只是在寻找一个快速的“一键”安装，检查顶层安装指南。如果您需要定制化安装，请根据参考部署使用指南一步一步地在单台系统上安装 StackStorm。

> **备注**
>
> 卢克，请使用源代码！我们努力保持文档是最新版本，但是要了解真正发生的事情最好的方法是查看安装脚本的代码。

- 系统安装要求
- 最小安装
  - ✓ 安装 libffi-devel 软件包
  - ✓ 调整 SELinux 策略
  - ✓ 安装依赖软件包
  - ✓ 创建软件仓库
  - ✓ 安装 StackStorm 组件
  - ✓ 创建机密 DataStore
  - ✓ 创建 Mistral 数据库
  - ✓ 配置 SSH 和 SUDO
  - ✓ 启动服务
  - ✓ 安装验证
- 配置用户认证
- 安装 WebUI，创建 SSL 终端
- 创建 ChatOps
- 安全注意事项
- 升级到 Extreme Workflow Composer

2.5.1 系统安装要求

请参考支持软件版本和系统要求。

2.5.2 最小安装

1）安装 libffi-devel 软件包

StackStorm 的依赖软件包 `libffi-devel` 可能没有随 RHEL 6 一起发布。在这种情况下，请按照 https://access.redhat.com/solutions/265523 的说明设置 `server-optional` 的软件仓库。或者找到与安装的 `libffi` 版本兼容的 `libffi-devel` 版本。例如：

```
[ec2-user@ip-172-30-0-79 ~]$ rpm -qa libffi
libffi-3.0.5-3.2.el6.x86_64

sudo yum localinstall -y ftp://rpmfind.net/linux/centos/6.9/os/x86_64/Packages/libffi-devel-3.0.5-3.2.el6.x86_64.rpm
```

使用如 http://rpmfind.net 软件包服务网站中可以找到需要的 RPM 包

2）调整 SELinux 策略

如果您的系统使用 SELinux 的 Enforcing 模式，请按照下面的说明调整 SELinux 策略。这是成功安装所必需的。如果您不想调整这些策略，您也可以根据您的安全实践对它们进行调整。

首先，请检查 SELinux 是否使用 Enforcing 模式：

```
getenforce
```

如果上面命令返回是'Enforcing'，那么执行下面命令：

```
# SELINUX management tools, not available for some minimal installations
sudo yum install -y policycoreutils-python

# Allow network access for nginx
sudo setsebool -P httpd_can_network_connect 1
```

备注

如果您看到诸如"SELinux: Could not downgrade policy file"之类的消息，这意味着 SELinux 是 disabled 模式，您正在尝试调整策略配置。您可以忽略这个错误。

3）安装依赖软件

备注

目前的首选和受支持的 MongoDB 是 3.4 版本。这也是安装脚本安装的版本。StackStorm v2.2.0 和更高版本支持 MongoDB3.4。StackStorm 的旧版本(在 V1.6.0 之前)只支持 MongoDB2.x。

安装 MongoDB、RabbitMQ、PostgreSQL 数据库：

```
sudo yum -y install https://dl.fedoraproject.org/pub/epel/epel-release-latest-6.noarch.rpm

# Add key and repo for the latest stable MongoDB (3.4)
sudo rpm --import https://www.mongodb.org/static/pgp/server-3.4.asc
sudo sh -c "cat <<EOT > /etc/yum.repos.d/mongodb-org-3.4.repo
[mongodb-org-3.4]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.4/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.4.asc
EOT"

sudo yum -y install mongodb-org
sudo yum -y install rabbitmq-server
sudo service mongod start
sudo service rabbitmq-server start
sudo chkconfig mongod on
sudo chkconfig rabbitmq-server on

# Install and configure postgres 9.4. Based on the OS type, install the ``redhat`` one or ``centos`` one.
# RHEL:
if grep -q "Red Hat" /etc/redhat-release; then sudo yum -y localinstall
http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-redhat94-9.4-2.noarch.rpm; fi

# CentOS:
if grep -q "CentOS" /etc/redhat-release; then sudo yum -y localinstall
http://yum.postgresql.org/9.4/redhat/rhel-6-x86_64/pgdg-centos94-9.4-2.noarch.rpm; fi

sudo yum -y install postgresql94-server postgresql94-contrib postgresql94-devel
```

```
# Initialize PostgreSQL
sudo service postgresql-9.4 initdb

# Make localhost connections to use an MD5-encrypted password for authentication
sudo sed -i "s/\(host.*all.*all.*127.0.0.1\/32.*\)ident/\1md5/" /var/lib/pgsql/9.4/data/pg_hba.conf
sudo sed -i "s/\(host.*all.*all.*::1\/128.*\)ident/\1md5/" /var/lib/pgsql/9.4/data/pg_hba.conf

# Start PostgreSQL service
sudo service postgresql-9.4 start
sudo chkconfig postgresql-9.4 on
```

4）创建软件仓库

下面的脚本将检测您的平台和体系结构，并设置合适的 StackStorm 软件仓库。还将添加用于包签名的 GPG 密钥。

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

5）安装 StackStorm 软件组件

```
sudo yum install -y st2 st2mistral
```

如果您没有在同一个系统上运行 RabbitMQ、MongoDB 或 PostgreSQL，或者更改了默认值，请调整以下参数设置：

- RabbitMQ 连接配置在`/etc/st2/st2.conf`和`/etc/mistral/mistral.conf`
- MongoDB 配置在`/etc/st2/st2.conf`
- PostgreSQL 配置在`/etc/mistral/mistral.conf`

详见配置文档章节。

6）创建加密 Datastore

键值对存储允许用户存储加密的值(秘密)。这些存储使用对称加密算法(AES256)。要生成密码密钥，请运行以下命令：

```
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
```

7）创建 Mistral 数据库

运行以下这些命令来创建 Mstral PostgreSQL 数据库：

```
# Create Mistral DB in PostgreSQL
cat << EHD | sudo -u postgres psql
CREATE ROLE mistral WITH CREATEDB LOGIN ENCRYPTED PASSWORD 'StackStorm';
CREATE DATABASE mistral OWNER mistral;
EHD

# Setup Mistral DB tables, etc.
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf upgrade head
# Register mistral actions
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf populate |
grep -v -e openstack -e keystone
```

8）配置 SSH 和 SUDO

要运行本地和远程 shell 的 action 操作，StackStorm 使用特定的系统用户(默认情况下为 `Stanley`)。对于远程 Linux 使用 SSH 进行操作。建议在所有远程主机上配置基于公钥的 SSH 访问。我们也建议将访问本地主机也配置成 SSH 访问，以便运行示例和进行测试。

- 创建 StackStorm 系统用户，设置成无密码的 Sudo，并将“本地主机”设置为 SSH 访问，以便于在本地测试基于 SSH 的操作。有些您可能也需要提升权限才能执行操作：

```
# 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
```

- 在远程主机上配置 SSH 访问并启用无密码的 sudo，StackStorm 将通过 SSH 执行远程的 action 操作。使用上一步生成的公钥，按照配置 SSH 中的说明进行操作。如要控制 Windows 服务器，需要配置 Windows 执行器的访问权限。
- 如果您正在使用其他用户或带路径的 SSH 密钥，则需要在 `/etc/st2/st2.conf` 中更改此部分：

```
[system_user]
user = stanley
ssh_key_file = /home/stanley/.ssh/stanley_rsa
```

9）启动服务

启动服务：

```
sudo st2ctl start
```

注册传感器、规则和 action 操作：

```
sudo st2ctl reload
```

10）安装验证

以下命令将测试您安装的 StackStorm。它们都应该成功地执行完成：

```
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
```

使用脚本管理 StackStorm 服务：

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

在这一点上，您有一个很小的安装工作，并可以愉快地玩 StackStorm：请按照快速入门教程、部署示例、探索和从 StackStorm Exchange 安装 pack 软件包。

但是还不能享受 WebUI 的乐趣，安全上也没有 SSL 或身份验证，也没有 ChatOps 的欢乐，没有钱也就没有 Extreme Workflow Composer。请继续往下读！

2.5.3 配置用户认证

为了简单起见，使用基于文件的身份验证的参考部署。请参考身份验证来配置，并使用 PAM 或 LDAP 后端的身份验证。

创建基于文件的身份认证：

- 创建用户、设置口令：

```
# Install htpasswd utility if you don't have it
sudo yum -y install httpd-tools
# Create a user record in a password file.
sudo htpasswd -bs /etc/st2/htpasswd st2admin 'Ch@ngeMe'
```

- 在中`/etc/st2/st2.conf`启用并配置身份认证：

```
[auth]
# ...
enable = True
backend = flat_file
backend_kwargs = {"file_path": "/etc/st2/htpasswd"}
# ...
```

- 重启 st2api 服务：

```
sudo st2ctl restart-component st2api
```

- 身份验证、设置环境变量的令牌，并检查它是否有效：

```
# Get an auth token to use in CLI or API
st2 auth st2admin
```

```
# A shortcut to authenticate and export the token
export ST2_AUTH_TOKEN=$(st2 auth st2admin -p 'Ch@ngeMe' -t)

# Check that it works
st2 action list
```

查看 CLI 参考指南，了解其他通过 CLI 进行身份验证的简便方法。

2.5.4 安装 WebUI，创建 SSL 终端

Nginx 用于服务 WebUI 静态文件、将 HTTP 重定向到 HTTPS、提供 SSL 终端以及反向代理 st2auth 和 st2api 的 API 端点。要设置它：需要安装 `st2web` 和 `nginx` 软件包，生成证书或将已有证书放在 `/etc/ssl/st2` 下，然后配置 StackStorm 站点的 nginx 服务的配置文件 st2.conf。

StackStorm 依赖于版本>=1.7.5 的 Nginx。在 RHEL 软件仓库中有一个旧版本，因此您需要添加官方的 Nginx 软件仓库：

```
# Add key and repo for the latest stable nginx
sudo rpm --import http://nginx.org/keys/nginx_signing.key
sudo sh -c "cat <<EOT > /etc/yum.repos.d/nginx.repo
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/rhel/$releasever/x86_64/
gpgcheck=1
enabled=1
EOT"

# Install nginx, avoid epel repo in favor of nginx.org
sudo yum --disablerepo='epel' install -y nginx

# Install st2web
sudo yum install -y st2web

# Generate a 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 365 -nodes -subj "/C=US/ST=California/L=Palo Alto/O=StackStorm/OU=Information \
Technology/CN=$(hostname)"

# Copy and enable the supplied nginx config file
sudo cp /usr/share/doc/st2/conf/nginx/st2.conf /etc/nginx/conf.d/
```

```
# Disable default_server configuration in existing /etc/nginx/nginx.conf
sudo sed -i 's/default_server//g' /etc/nginx/conf.d/default.conf

sudo service nginx restart
sudo chkconfig nginx on
```

如果您需要修改 nginx 配置中的端口或 url 路径，那么需要在配置文件 `/opt/stackstorm/static/webui/config.js` 上对 st2web 配置选项进行相应的更改。

用您的浏览器访问 `https://${ST2_HOSTNAME}`，登录 WebUI。

如果您正在要从服务器外部访问 API，并且按照以下说明配置了 nginx，请使用 `https://${EXTERNAL_IP}/api/v1/${REST_ENDPOINT}`。

例如：

```
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
```

类似地，您可以使用 `https://${EXTERNAL_IP}/auth/v1/${AUTH_ENDPOINT}` 连接到认证的 REST 端点。

通过向相应资源的 CLI 命令添加 `--debug` 调试选项，可以看到资源的实际 REST 端点。

例如，要查看获取端点的 action 操作，请调用：

```
st2 --debug action list
```

2.5.5 创建 ChatOps

如果您已经运行了 Hubot 实例，您可以如下所述方式安装 hubot-stackstorm plugin 插件并配置 StackStorm 的环境变量。否则，启用 StackStorm ChatOps 最简单的方法是使用 st2chatops 软件包。

- ❑ 验证是否安装了 `chatops` 软件包，并启用通知规则：

```
# Ensure chatops pack is in place
ls /opt/stackstorm/packs/chatops
# Create notification rule if not yet enabled
st2 rule get chatops.notify || st2 rule create /opt/stackstorm/packs/chatops/rules/notify_hubot.yaml
```

- ❑ 添加 NodeJS v6 软件仓库

```
curl -sL https://rpm.nodesource.com/setup_6.x | sudo -E bash -
```

- ❑ 安装 `st2chatops` 软件包

```
sudo yum install -y st2chatops
```

- 检查和编辑`/opt/stackstorm/chatops/st2chatops.env`配置文件，将其指向您正在使用的 StackStorm 安装和聊天服务。至少，您应该生成一个 API 密钥并设置`ST2_API_KEY`变量。默认情况下，`st2api`和`st2auth`应该位于同一个主机上。如果不是这样，请更新`ST2_API`和`ST2_Auth_URL`变量，或者用`ST2_HOSTNAME`指向正确的主机。

使用了 Slack 示例配置。若要创建这个配置，请转到 Slack 的 Web 管理页面，创建一个 BOT、并将身份验证令牌复制到`HUBOT_SLACK_TOKEN`中。

如果您使用的是不同的聊天服务，请在`st2chatops.env`中：Slack、HipChat、Yammer、Flowdock、IRC 、XMPP 中的`Chat service adapter settings`部分下设置相应的环境变量。

启动服务：

```
sudo service st2chatops start

# Ensure it will start on boot
sudo chkconfig st2chatops on
```

重新装入 st2 的 pack 软件包，确保`chatops.notify`规则已经注册：

```
sudo st2ctl reload --register-all
```

就这样，到你的聊天室开始聊天。在 ChatOps 部分阅读更多内容。

2.5.6 安全注意事项

默认情况下，在安装了 MongoDB、RabbitMQ 和 PostgreSQL 后，它们禁用身份验证或使用默认静态密码。因此，在安装完成这些服务之后，您应该调整配置，使用增强随机生成的密码作为身份验证。

注意：如果您使用 StackStorm 安装脚本，将会自动为您完成。

为这些服务配置授权和密码超出了本文档的范围。有关更多信息，请参阅以下链接：

- MongoDB - https://docs.mongodb.com/manual/tutorial/enable-authentication/, https://docs.mongodb.com/manual/core/authorization/
- RabbitMQ - https://www.rabbitmq.com/authentication.html
- PostgreSQL - https://www.postgresql.org/docs/9.4/static/auth-methods.html

启用这些组件的身份验证之后，还需要重启 StackStorm 服务，使新设置生效。

这意味着修改下面配置选项：

1）StackStorm - `/etc/st2/st2.conf`

- `database.username` – MongoDB 数据库用户名
- `database.password` – MongoDB 数据库口令
- `messaging.url` – RabbitMQ 传输 url(`amqp://<username>:<password>@<hostname>:5672`)

2）Mistral - `/etc/mistral/mistral.conf`

- `database.connection` – PostgreSQL 数据库连接字符串 (`postgresql+psycopg2://<username>:<password>@<hostname>/mistral`)
- `transport_url` – RabbitMQ 传输 url (`rabbit://<username>:<password>@<hostname>:5672`)

此外，强烈鼓励您按照以下这些运行网络服务的最佳实践：

- 确保服务之间的通信是加密的。为 MongoDB、RabbitMQ 和 PostgreSQL 之间启用 SSL/TLS。
- 将服务配置为只侦听本地主机，并在需要时侦听内部 IP 地址。通常 StackStorm(MongoDB、RabbitMQ、PostgreSQL) 大多数服务不需要使用在公共上可用 IP 地址。
- 配置防火墙并设置白名单。防火墙应该只允许那些需要访问这些服务的用户和系统可以访问。用户通常需要对 API 和 auth 服务进行访问，但其他依赖服务，如 MongoDB、RabbitMQ 和 PostgreSQL 则不需要访问。用户不应直接访问这些组件，而只允许 StackStorm 组件与他们通信。
- 在可能的情况下，您还应该利用其他基于网络的隔离和安全特性，如 DMZ。

上面提到的步骤对于在多个服务器上运行 StackStorm 组件的分布式生产部署特别重要。

2.5.7 升级到 Extreme Workflow Composer

Extreme Workflow Composer 将工作流设计器(用于工作流创建/编辑的图形工具)、RBAC 和 LDAP 添加到 StackStorm 中。它作为一组附加的软件包部署在 StackStorm 之上。您将需要一个 Extreme Workflow Composer 的激活的订阅和一个许可证密钥来访问 Extreme Workflow Composer 的存储仓库。

To learn more about Extreme Workflow Composer, request a quote, or get an evaluation license go to stackstorm.com/product.

要了解有关 Extreme Workflow Composer 的更多信息，查询报价或获得评估许可，请访问 stackstorm.com/product。

要安装 Extreme Workflow Composer，请将下面命令中的`${EWC_LICENSE_KEY}`替换为注册或购买时收到的密钥，并执行以下命令：

```
# Set up Extreme Workflow Composer repository access
curl -s
https://${EWC_LICENSE_KEY}:@packagecloud.io/install/repositories/StackStorm/enterprise/script.rpm.sh | sudo bash
# Install Extreme Workflow Composer
sudo yum install -y bwc-enterprise
```

下一章节内容

- ❑ 查看快速入门指南创建简单的自动化。
- ❑ 了解更多的 action、trigger、rule
  - ✓ 按照 Packs 指南，从 StackStorm Exchange 安装集成的 pack 软件包
  - ✓ 把您的脚本转成 StackStorm 的 action；
  - ✓ 学习如何编写自定义 action；
- ❑ 使用工作流将 action 整合到更高级别的自动化中—工作流 Workflows
- ❑ 查看 stackstorm.com 上的教程

2.6 Docker 环境安装

https://docs.stackstorm.com/install/docker.html

您喜欢 Docker 和 Kubernetes 吗？我们也喜欢！

让 StackStorm 运行的最快速的方法之一是使用 Docker。本章节将向您介绍如何使用 Docker 方式部署 StackStorm 的基础知识。

有关更多详细信息和示例，请查阅 st2-docker GitHub repo 的 README 文件。

2.6.1 主机要求

- ❑ 安装最新版本的 Docker 引擎，并可选择 `docker-compose`。详细安装说明见 https://www.docker.com/community-edition 和 https://docs.docker.com/compose/install。

备注

We require at least version 1.13.0 of Docker engine. If you choose to use `docker-compose` it must also be at least version 1.13.0.

最少需要 1.13.0 版本的 Docker 引擎。如果选择了 `docker-compose`，也必须至少是 1.13.0 的版本。

- ❑ 如果使用 Kubernetes，更多信息请查阅 Kubernetes README 文件。

2.6.2 Docker 镜像

`stackstorm/stackstorm` 镜像中预装了 `st2`、`st2web`、`st2mistral`、`st2chatops` 等软件包。

我们使用版本标签，所以如果您安装了 `stackstorm/stackstorm:2.3.2` 镜像，那么它就是 StackStorm 2.3.2 发行版。类似地，如果您安装的是 `stackstorm/stackstorm:2.2.1`，那么它就是 StackStorm 2.2.1 发行版。`stackstorm/stackstorm:latest` 镜像简单的引用版本号最高的镜像。别担心，这仍然是一个稳定的 GA 版本，而不是一个预发布版本。

mongo、rabbitmq、postgres 和 redis 容器将它们的数据存储在持久存储上。此外，stackstorm 容器保存 `/var/log` 的内容。如果您不希望持久化这些数据，那么从 `docker-compose.yml` 中删除相应的配置项。

2.6.3 使用 Docker

如果 Docker 引擎和 `docker-compose` 正确安装之后，就很容易启动了。

首先，克隆 `st2-docker` 仓库并将目录更改为 `st2-docker`。除非另有说明，否则所有后续指令都假定它们在 `st2-docker` 目录中运行：

```
git clone https://github.com/stackstorm/st2-docker
cd st2-docker
```

其次，执行：

```
make env
```

在 `conf/` 目录下创建 `docker-compose` 使用的环境文件。在做这之前，您可能根据需要更改一些变量。如果您不使用任何集群服务（例如 mongo、redis、postgres、rabbitmq），默认值应该是可以的。

下面是可用于定制容器的可用选配置项的完整清单：

Parameter	Description
`ST2_USER`	StackStorm 帐号用户名
`ST2_PASSWORD`	StackStorm 帐号口令
`MONGO_HOST`	MongoDB 服务器主机名
`MONGO_PORT`	MongoDB 服务端口（典型是 27017）
`MONGO_DB`	（可选）MongoDB 数据库名，不指定将使用 st2
`MONGO_USER`	（可选）MongoDB 用户名，如果没有使用证书且没有设置 *MONGO_PASS，就会使用这个*
`MONGO_PASS`	（可选）MongoDB 口令
`RABBITMQ_HOST`	RabbitMQ 服务主机名
`RABBITMQ_PORT`	RabbitMQ 服务端口（典型是 *5672*）
`RABBITMQ_DEFAULT_USER`	RabbitMQ 用户名
`RABBITMQ_DEFAULT_PASS`	RabbitMQ 口令
`POSTGRES_HOST`	PostgreSQL 服务主机名
`POSTGRES_PORT`	PostgreSQL 服务端口（典型是 *5432*）
`POSTGRES_DB`	PostgreSQL 数据库名称

Parameter	Description
POSTGRES_USER	PostgreSQL 用户名
POSTGRES_PASSWORD	PostgreSQL 口令
REDIS_HOST	Redis 服务主机名
REDIS_PORT	Redis 服务端口
REDIS_PASSWORD	（可选）Redis 口令

第三，启动容器

```
docker-compose up -d
```

这将从 Docker Hub 中拉取所需的镜像，然后启动它们

停止容器，执行：

```
docker-compose down
```

2.7 Ansible Playbooks 方式安装

https://docs.stackstorm.com/install/ansible.html

要使用 Ansible 部署 StackStorm 吗？不要再看了，这是有关使用 Ansible 剧本和角色安装 StackStorm 的详细指南。完美的可重复、可配置和等幂性的生产环境友好的 StackStorm 安装方式。

我们这里使用的所有 playbook 源码都可以在 GitHub 仓库：StackStorm/ansible-st2 中下载。

内容目录

2.7.1 支持的平台

我们的 Ansible playbooks 手动安装方式支持相同的平台，即：

- Ubuntu Trusty (14.04)
- Ubuntu Xenial (16.04)
- RHEL 6/CentOS 6
- RHEL 7/CentOS 7

也是同样的系统要求。

2.7.2 快速入门

要从使用单个节点部署和默认配置方式开始，请执行以下命令：

```
git clone https://github.com/StackStorm/ansible-st2.git
cd ansible-st2

ansible-playbook stackstorm.yml
```

备注

请记住，默认的 StackStorm 凭证是按照 https://github.com/StackStorm/ansible-st2#variables 中的 `testu:testp`，别忘了修改成更安全的配置。

2.7.3 Roles

在 `stackstorm.yml` 剧本场景中包含了以下 Ansible `roles` 完成安装：

- `epel` - `RHEL/CentOS` 附加软件包的仓库
- `mongodb` – 主要数据库存储引擎
- `rabbitmq` - 消息代理
- `postgresql` – Mistral 的数据库引擎
- `st2repos` – 添加 StackStorm 的 PackageCloud 仓库
- `st2` - 安装、配置 StackStorm 软件自身

- `st2mistral` - 安装、配置支持 StackStorm Mistral 工作量引擎
- `nginx` - `st2web` 的依赖软件
- `st2web` – StackStorm 的 Nice & shiny Web UI
- `nodejs` - `st2chatops` 的依赖软件
- `st2chatops` - 安装、配置支持与 StackStorm 集成的 hubot 适配器 st2chatops
- `st2smoketests` - 简单检查判断 StackStorm 是否工作
- `bwc` – 安装、配置包括 `LDAP` 和 `RBAC 的` Extreme Workflow Composer
- `bwc_smoketests` – 简单检查判断 Extreme Workflow Composer 是否工作

2.7.4 Play 实例

下面是一个更高级的示例，演示如何自定义部署 StackStorm：

```
- name: Install StackStorm with all services on a single node
  hosts: all
  roles:
    - mongodb
    - rabbitmq
    - postgresql
    - nginx
    - nodejs

    - name: Install StackStorm Packagecloud repository
      role: st2repo
      vars:
        st2repo_name: stable

    - name: Install and configure st2
      role: st2
      vars:
        st2_version: latest
        st2_auth_enable: yes
        st2_auth_username: testu
        st2_auth_password: testp
        st2_save_credentials: yes
        st2_system_user: stanley
        st2_system_user_in_sudoers: yes
        # Dict to edit https://github.com/StackStorm/st2/blob/master/conf/st2.conf.sample
        st2_config: {}
```

```
    - name: Install and configure st2mistral
      role: st2mistral
      vars:
        st2mistral_version: latest
        st2mistral_db: mistral
        st2mistral_db_username: mistral
        st2mistral_db_password: StackStorm
        # Dict to edit https://github.com/StackStorm/st2-packages/blob/master/packages/st2mistral/conf/mistral.conf
        st2mistral_config: {}

    - name: Install st2web
      role: st2web

    - name: Install st2chatops with "slack" hubot adapter
      role: st2chatops
      vars:
        st2chatops_version: latest
        st2chatops_st2_api_key: CHANGE-ME-PLEASE # (optional) This can be generated using "st2 apikey create -k"
        st2chatops_hubot_adapter: slack
        st2chatops_config:
          HUBOT_SLACK_TOKEN: xoxb-CHANGE-ME-PLEASE

    - name: Verify StackStorm Installation
      role: st2smoketests
```

下载完整的变量清单。

2.7.5 定制 st2web 的 SSL 证书

默认情况下，我们为 `st2web` 角色中的 `nginx` 生成一个自签名证书。如果您有自己的签名证书，则可以使用该证书：

```
- name: Configure st2web with custom SSL certificate
  role: st2web
  vars:
    st2web_ssl_certificate: "{{ lookup('file', 'local/path/to/domain-name.crt') }}"
    st2web_ssl_certificate_key: "{{ lookup('file', 'local/path/to/domain-name.key') }}"
```

2.7.6 后台代理的安装方式

如果您需要从后台代理方式安装，则可以使用环境变量 `http_proxy`、`https_proxy`、`no_proxy`。他们将在执行过程中直接通过代理。

```
---
- name: Install st2
```

```
  hosts: all
  environment:
    http_proxy: http://proxy.example.net:8080
    https_proxy: http://proxy.example.net:8080
    no_proxy: 127.0.0.1,localhost
  roles:
    - st2
```

2.7.7 Extreme Workflow Composer

下面例子中演示如何添加具有 LDAP 身份验证和 RBAC 配置的 Extreme Workflow Composer，以允许/限制/定制特定用户的 StackStorm 功能：

```
- name: Install StackStorm Enterprise
  hosts: all
  roles:
    - name: Install and configure StackStorm Enterprise (EWC)
      role: bwc
      vars:
        bwc_repo: enterprise
        bwc_license: CHANGE-ME-PLEASE
        bwc_version: latest
        # Configure LDAP backend
        # See: https://ewc-docs.extremenetworks.com/authentication.html#ldap
        bwc_ldap:
          backend_kwargs:
            bind_dn: "cn=Administrator,cn=users,dc=change-you-org,dc=net"
            bind_password: "foobar123"
            base_ou: "dc=example,dc=net"
            group_dns:
              - "CN=stormers,OU=groups,DC=example,DC=net"
            host: identity.example.net
            port: 389
            id_attr: "samAccountName"
        # Configure RBAC
        # See: https://ewc-docs.extremenetworks.com/rbac.html
        bwc_rbac:
          # Define EWC roles and permissions
          # https://ewc-docs.extremenetworks.com/rbac.html#defining-roles-and-permission-grants
          roles:
            - name: core_local_only
              description: "This role has access only to action core.local in pack 'core'"
              enabled: true
              permission_grants:
```

```
                - resource_uid: "action:core:local"
                  permission_types:
                    - action_execute
                    - action_view
                - permission_types:
                  - runner_type_list
            # Assign roles to specific users
            # https://ewc-docs.extremenetworks.com/rbac.html#defining-user-role-assignments
            assignments:
              - name: test_user
                roles:
                  - core_local_only
              - name: stanley
                roles:
                  - admin
              - name: chuck_norris
                roles:
                  - system_admin

    - name: Verify EWC Installation
      role: bwc_smoketests
```

备注

有关更新和更详细的示例、说明和代码，请参阅 https://github.com/StackStorm/ansible-st2。如果您熟悉 Ansible，发现了一个 bug，或者想提交一个特性或请求，非常欢迎您的贡献！

2.8 安装 Extreme Workflow Composer

https://docs.stackstorm.com/install/bwc.html

StackStorm 是一个事件驱动的 DevOps 自动化平台，具有适合小型企业和团队的所有基本功能。在 Apache 2.0 许可下，它是免费的开源代码。

Extreme Workflow Composer(EWC)是 StackStorm 自动化平台的商业版本。EWC 增加了优先支持服务，如精细的访问控制、LDAP 和 Workflow Designer 等高级功能。可以获得评估板授权、报价申请等，掌握更多有关

Extreme Workflow Composer 信息，请访问 extremenetworks.com/product/workflow-composer

您还可以在 Extreme Workflow Composer 系统之上再增加网络自动化套件（Network Automation Suites），更多参见 ewc-docs.extremenetworks.com/solutions/overview.html。

为了快速安装 Extreme Workflow Composer 评估板，在一个干净的 64 位 Linux 系统（见系统要求）上执行下面命令，把下面`${EWC_LICENSE_KEY}`替换成您收到的评估板或采购 EWC 的授权码。

```
curl -sSL -O https://stackstorm.com/bwc/install.sh && chmod +x install.sh
./install.sh --user=st2admin --password='Ch@ngeMe' --license=${EWC_LICENSE_KEY}
```

已经有了一个工作的 StackStorm 系统，想要添加 Extreme Workflow Composer？没问题！不需要重新安装新系统。只需要在现有系统上安装 Extreme Workflow Composer，再次用注册时收到的授权许可替换`${EWC_License_Key}`。

- ❑ Ubuntu 系统

```
# Set up Extreme Workflow Composer repository access
curl -s
https://${EWC_LICENSE_KEY}:@packagecloud.io/install/repositories/StackStorm/enterprise/script.deb.sh | sudo bash
# Install Extreme Workflow Composer
sudo apt-get install -y bwc-enterprise
```

- ❑ RedHat/CentOS 系统

```
# Set up Extreme Workflow Composer repository access
curl -s
https://${EWC_LICENSE_KEY}:@packagecloud.io/install/repositories/StackStorm/enterprise/script.rpm.sh | sudo bash
# Install Extreme Workflow Composer
sudo yum install -y bwc-enterprise
```

要了解安装过程的全部详细信息，或手动安装 Extreme Workflow Composer，请按照您的 Linux 版本的选择安装指南：Ubuntu Trusty/Xenial、RHEL 7/CentOS 7、RHEL 6/CentOS 6。它将指导您安装和配置 StackStorm

和 Extreme Workflow Composer。最后一步就是是“升级到 Extreme Workflow Composer”。

提问？问题？建议？都很欢迎！

- 支持论坛（Support Forum）
- Slack 社区频道：stackstorm-community.slack.com (注册 这里)
- 支持邮箱：support@stackstorm.com

2.9 详细配置

https://docs.stackstorm.com/install/config/index.html

本节提供了关于 StackStorm 配置选项的详细信息。

StackStorm 配置文件位于/etc/st2/st2.conf。实例中包含所有配置选项的配置文件可参考 st2.conf.sample。

- Nginx 和 WSGI
- 配置 MongoDB
- 配置 RabbitMQ
- 配置 SSH
- SUDO 访问
- 配置 Logging
- 配置 Mistral
- 授权认证
- 配置 ChatOps
- 配置屏蔽私密信息
- Web UI
- 配置 Windows 的 Runners

提问？问题？建议？都很欢迎！

- 支持论坛（Support Forum）
- Slack 社区频道：stackstorm-community.slack.com (注册 这里)

- 支持邮箱：support@stackstorm.com

2.9.1 Nginx 与 WSGI

https://docs.stackstorm.com/install/config/config.html

生产环境中，StackStorm 安装使用 nginx 作为 SSL 终端，提供 WebUI 静态内容，并通过 gunicorn/uwsgi 作为 WSGI 应用程序运行 st2auth 和 st2api。StackStormnginx 配置可以在`/etc/nginx/sites-enabled/st2*.conf`上找到。

`st2auth`和`st2api`也可以使用内置的简单 Python 服务器运行。这是用于发展，并强烈反对任何生产。请注意，`/etc/st2.conf`中的某些设置只有在开发模式下运行时才有效，而在 WSGI 服务器下运行时则不适用。请参阅 st2.conf.sample 中的注释。

2.9.2 配置 MongoDB

StackStorm 需要连接到 MongoDB 才可以操作。

在/etc/st2/st2.conf 中包括下面配置

```
[database]
host = <MongoDB host>
port = <MongoDB server port>
db_name = <User define database name, usually st2>
username = <username for db login>
password = <password for db login>
```

`username`和 `password`配置项是可选的。

StackStorm 也可以用 MongoDB URI string 支持 MongoDB replica sets。

/etc/st2/st2.conf 中包括下面配置

```
[database]
host = mongodb://<#MDB_NODE_1>,<#MDB_NODE_2>,<#MDB_NODE_3>/?replicaSet
=<#MDB_REPLICA_SET_NAME>
```

- 你也能够在连接字符串中增加端口、用户名、口令等，详见 https://docs.mongodb.com/v3.4/reference/connection-string/
- 要更多了解如何创建 MongoDB 复制的配置，详见 https://docs.mongodb.com/v3.4/tutorial/deploy-replica-set/

StackStorm 也支持 SSL/TLS 的加密连接，在上面列出的配置之外还需要配置一些属性。

在/etc/st2/st2.conf 中包括下面配置

```
[database]
...
ssl = <True or False>
ssl_keyfile = <Path to key file>
ssl_certfile = <Path to certificate>
ssl_cert_reqs = <One of none, optional or required>
ssl_ca_certs = <Path to certificate form mongod>
ssl_match_hostname = <True or False>
```

- `ssl` – 启用或禁用 TLS/SSL 上的连接。默认是禁用 False
- `ssl_keyfile` - 用于标识针对 MongoDB 的本地连接的私有密钥文件。如果指定 SSL，则假定为 True。
- `ssl_certfile` - 用于标识本地连接的证书文件。如果指定了 ssl 证书文件，则假定为 True。
- `ssl_cert_reqs` - 指定从连接的另一侧是否需要证书，以及是否需要验证。
- `ssl_ca_certs` - 包含一组连接 CA 证书的证书文件，用于验证从 MongoDB 传递的证书。
- `ssl_match_hostname` - 启用或禁用主机名匹配。不建议禁用，默认为 True。

备注

只有某些 MongoDB 发行版支持 SSL/TLS：

MongoDB 企业版能够支持 SSL/TLS。

从源代码构建的 MongoDB，可以支持 SSL/TLS 连接。更多详细信息见 https://github.com/mongodb/mongo/wiki/Build-Mongodb-From-Source。

2.9.3 配置 RabbitMQ

StackStorm 在其服务之间传递消息使用的是 RabbitMQ。

在配置文件/etc/st2/st2.conf 中包含下面内容：

```
[messaging]
url = amqp://#RMQ_USER:#RMQ_PASSWD@#RMQ_HOST:#RMQ_PORT/#RMQ_VHOST
```

其中`#RMQ_VHOST` 属性是可选的，也可以是空白。

StackStorm 也支持 RabbitMQ 集群。

在配置文件/etc/st2/st2.conf 中包含下面内容：

```
[messaging]
cluster_urls =
amqp://#RMQ_USER:#RMQ_PASSWD@#RMQ_NODE_1:#RMQ_PORT/#RMQ_VHOST,
amqp://#RMQ_USER:#RMQ_PASSWD@#RMQ_NODE_2:#RMQ_PORT/#RMQ_VHOST,
amqp://#RMQ_USER:#RMQ_PASSWD@#RMQ_NODE_3:#RMQ_PORT/#RMQ_VHOST
```

- 要更多了解如何创建 RabbitMQ 集群，详见 https://www.rabbitmq.com/clustering.html。
- RabbitMQ 高可用指南，详见 https://www.rabbitmq.com/ha.html

2.9.4 配置 SSH

要在远程主机上运行 action 操作，StackStorm 将使用 SSH。建议在所有远程主机上使用基于公钥的 SSH 访问。

StackStorm 的 ssh 用户和 SSH 密钥的路径信息将在`/etc/st2/st2.conf`中设置。在安装过程中，一键安装脚本为本地服务器上配置了 ssh，用户为`stanley`。

按照以下步骤配置远程系统上的用户 `stanley`：

```
useradd stanley
mkdir -p /home/stanley/.ssh
chmod 0700 /home/stanley/.ssh

# generate ssh keys and copy over public key to remote box.
ssh-keygen -f /home/stanley/.ssh/stanley_rsa -P ""
cp ${KEY_LOCATION}/stanley_rsa.pub /home/stanley/.ssh/stanley_rsa.pub

# authorize key-based access.
cat /home/stanley/.ssh/stanley_rsa.pub >> /home/stanley/.ssh/authorized_keys
chmod 0600 /home/stanley/.ssh/authorized_keys
chown -R stanley:stanley /home/stanley
echo "stanley       ALL=(ALL)           NOPASSWD: SETENV: ALL" >> /etc/sudoers.d/st2

# ensure requiretty is not set to default in the /etc/sudoers file.
sudo sed -i -r "s/^Defaults\s+\+requiretty/# Defaults +requiretty/g" /etc/sudoers
```

请在 StackStorm 系统中执行以下操作进行验证：

```
# ssh should not require a password since the key is already provided
ssh -i /home/stanley/.ssh/stanley_rsa stanely@host.example.com

# make sure that no password is required
```

```
sudo su
```

1）SSH 排错

验证无密码 SSH 配置是否适用于目标。假设默认用户 `stanley`：

```
sudo ssh -i /home/stanley/.ssh/stanley_rsa -t stanley@host.example.com uname -a
```

2）用 SSH 配置

StackStorm 允许向系统用户加载本地的 SSH 配置文件。这是一个可配置的选项。若要启用，请将以下内容添加到`/etc/st2/st2.conf`

```
[ssh_runner]
use_ssh_config = True
...
```

2.9.5 SUDO 访问

StackStorm 的 `shell` 操作由特定用户执行，包括 `local-shell-cmd`、`local-shell-script`、`remote-shell-cmd`、`remote-shell-script`。确实情况下，这个用户是 `stanley`，可以在 st2.conf 中进行配置。

备注

用户 `stanley` 需要下面权限：

- Sudo 可以访问所有需要运行脚本操作的服务器。
- 需要为所有命令设置 SETENV 选项。这样，当用户在不同用户下或以根权限执行本地执行程序操作时，本地执行程序操作可用的环境变量也是可用的。
- 由于某些 action 操作需要 sudo 特权，可以设置没有密码的 sudo 访问所有服务器。

一种设置无密码 sudo 的方法是对每个远程服务器执行以下操作：

```
echo "stanley      ALL=(ALL)          NOPASSWD: SETENV: ALL" >> /etc/sudoers.d/st2
```

2.9.6 配置日志

默认情况下，日志可以在`/var/log/st2`中找到。

- 使用标准日志设置，您可以在日志文件夹中看到类似 `st2*.log` 和 `st2*.audit.log` 文件。

- 每个组件的日志配置可以在`/etc/st2/logging.<component>.conf`中找到。这些文件使用 Python 日志配置格式（Python logging configuration format）。可以在这些配置文件中修改日志文件位置和其他配置项，例如将输出更改为使用 syslog。
- StackStorm 附带了配置文件示例，用于说明如何使用 syslog，它们位于`/etc/st2/syslog.<component>.conf`。要使用请编辑`/etc/st2/st2.conf`，并将`logging =`行更改为指向 syslog 配置文件。您还可以在 exchange-misc/syslog 上了解更多的说明和配置示例。
- 默认情况下，滚动日志是通过 logrotate 来处理的。默认滚动日志配置(logrotate.conf)包含在所有基于包的安装中。请注意，默认情况下在`/etc/st2*/logging.conf`中使用`handlers.RotatingFileHandler`，如果没有指定`maxBytes`和`backupCount`，默认情况下不执行任何滚动操作。如果您想让 Python 服务而不是 logrotate 来处理滚动日志，可以修改下面日志配置项：

```
[handler_fileHandler]
class=handlers.RotatingFileHandler
level=DEBUG
formatter=verboseConsoleFormatter
args=("logs/st2api.log", "a", 100000000, 5)
```

在这种情况下，当日志文件达到 100,000,000 字节(100 MB)时，日志文件将会滚动，最多保留 5 个旧日志文件。有关更多信息，请参见 RotatingFileHandler。

请记住，日志级别的名称需要大写(例如`DEBUG`、`INFO`等)。

- 传感器在自己的过程中运行，因此建议不要让传感器共享同一个`RotatingFileHandler`。要为每个传感器`/etc/st2reactor/logging.sensorcontainer.conf`配置一个单独的处理程序，可以按以下方式更新，其中`MySensor`是`mypack`包中的传感器，它将有自己的日志文件：

```
[loggers]
keys=root,MySensor

[handlers]
```

```
keys=consoleHandler, fileHandler, auditHandler, MySensorFileHandler, MySensorAuditHandler

[logger_MySensor]
level=INFO
handlers=consoleHandler, MySensorFileHandler, MySensorAuditHandler
propagate=0
qualname=st2.SensorWrapper.mypack.MySensor

[handler_MySensorFileHandler]
class=handlers.RotatingFileHandler
level=INFO
formatter=verboseConsoleFormatter
args=("logs/mysensor.log",)

[handler_vSphereEventSensorAuditHandler]
class=handlers.RotatingFileHandler
level=AUDIT
formatter=gelfFormatter
args=("logs/mysensor.audit.log",)
```

- 检查 LogStash 配置和 Kibana 仪表板，可以在 exchange-misc/logstash 查看漂亮的日志记录和审计。

2.9.7 配置 Mistral

在`/etc/st2/st2.conf`中的 Mistral 部分中有许多可配置的选项。如果没有 Mistral 部分的配置，则将使用默认值。默认情况下，所有与 Keystone 相关的选项都是未设置的，StackStorm 将不会允许任何用于身份验证的凭据传递给 Mistral。有关 Keystone 设置，请参阅 OpenStack 和 Mistral 的安装文档。

Options 选项	Description 描述
v2_base_url	Mistral API v2 的根端点。
retry_exp_msec	用于指数回退的成绩
retry_exp_max_msec	每次退出的最大时间。
retry_stop_max_msec	最多尝试次数。
keystone_username	OpenStack Keystone 的用户身份认证。
keystone_password	OpenStack Keystone 的口令身份认证。
keystone_project_name	OpenStack 项目范围。
keystone_auth_url	v3 Auth URL 的 OpenStack Keystone。

```
# Example with basic options. The v2_base_url is set to http://workflow.example.com:8989/v2.
# On connection error, the following configuration sets up the action runner to retry
# connecting to Mistral for up to 10 minutes. The retries is setup to be exponential for
# 5 minutes. So in this case, there will be two sets of exponential retries during
# the 10 minutes.

[mistral]
v2_base_url = http://workflow.example.com:8989/v2
retry_exp_msec = 1000
retry_exp_max_msec = 300000
retry_stop_max_msec = 600000
# Example with auth options.

[mistral]
v2_base_url = http://workflow.example.com:8989/v2
retry_exp_msec = 1000
retry_exp_max_msec = 300000
retry_stop_max_msec = 600000
keystone_username = mistral
keystone_password = pass123
keystone_project_name = default
keystone_auth_url = http://identity.example.com:5000/v3
```

2.9.8 身份验证

请参考“身份验证”，了解有关身份验证、与各种身份提供集成和管理 API 令牌的详细信息。

2.9.9 配置 ChatOps

StackStorm 提供了本地的双向 ChatOps 支持。要了解更多关于 ChatOps 的信息，以及如何手动配置，请参阅配置那节中的 ChatOps。

2.9.10 配置秘密掩蔽

为了在系统全局的基础上管理秘密掩蔽，您可以通过修改 `/etc/st2/st2.conf`，并在第二个 2 层级上控制秘密掩码。例如 API 和日志。请注意，此功能仅控制外部可见性的秘密，并且不控制那些由 StackStorm 管理的机密。

- 为了掩蔽从 API 响应秘密，这是在每个 API 的基础上启用的，并且只有管理用户可以使用。

```
[api]
...
mask_secrets = True
```

❑ 在日志中配置掩蔽秘密

```
[log]
...
mask_secrets = True
```

更多有关秘密掩蔽的信息和限制，请参阅秘密掩蔽。

2.9.11 Web UI

https://docs.stackstorm.com/install/config/webui.html

st2web 是一个基于 Angular 的 HTML5 Web 应用程序。它允许你控制执行的整个过程，从运行一个 action 到查看执行结果。还能帮助您浏览工作流的执行到单个任务的结果。而且都是实时的。

1）软件部署

st2web 是通过安装 st2web 的 rpm 或 deb 软件包。在使用单行安装方法时，默认情况下会使用此安装方法。您在将浏览器中访问 `https://<serverhostname>/` 访问 UI。如果您使用 st2vagrant，则为 https://192.168.16.20/。

st2web 是一个纯 HTML 5 应用程序，只包含 js 脚本、html 模板、CSS 样式和包括自定义字体和 SVG 图像的静态文件。使应用程序能正确工作，所有文件都应发送给浏览器。默认情况下，它们由 nginx 提供。也可以使用 Apache 或类似的专用 Web 服务器来个性化部署。

注意，StackStorm API 端点应该可以从浏览器访问，而不是运行静态内容的 Web 服务器。

2）参数配置

为了使 UI 正常工作，应该对 Web 客户端和 StackStorm 服务器端进行正确的配置。

在 Web 客户端，项目中的文件 `config.js` 包含 UI 可以连接到的服务器列

表。这文件通常在`/opt/stackstorm/Static/WebUI/config.js`。该文件由一个对象组组成，每个对象都具有`name`、`url`和`auth`属性。

```
hosts: [{
   name: 'Express Deployment',
   url: 'http://172.168.90.50:9101',
   auth: true
},{
   name: 'Development Environment',
   url: 'http://:9101'
   auth: 'https://:9100'
}]
```

可以为用户配置多个服务器以供选择。若要断开当前服务器的连接并返回到登录屏幕，请从 UI 右上角的下拉列表中选择“断开连接”。

在 StackStorm 端，也要对 CORS 配置。在`st2.conf`中，`[api]`节的`allow_origin`应该包含浏览器在每个请求中发送的 Origin Header。例如，如果您在自己的服务器上部署了 UI，并使用`http: //webui.example.com`访问它，您的配置应该如下所示：

```
[api]
# Host and port to bind the API server.
host = 0.0.0.0
port = 9101
logging = st2api/conf/logging.conf
# List of allowed origins for CORS, use when deploying st2web client
# The URL must match the one the browser uses to access the st2web
allow_origin = http://webui.example.com
```

Origin Header 包含结构、主机名和端口(如果不是 80)，但路径(包括最后的斜杠)应省略。

请注意，某些 Origin Header 已默认包括了，不需要额外的配置：

- `http://localhost:3000` – 部署`gulp`本地运行的版本
- `http://localhost:9101,http://127.0.0.1:9101` - `st2api` Pecan 框架部署
- `api_url`在`st2.conf`文件中的`[auth]`节中配置

虽然不建议这样做，并且会破坏您的安全性，但是您可以通过设置`allow_origin = *`允许每个 Web UI 都可以访问到您的服务器。

3）身份认证

要将 st2web 配置为支持身份验证，需要编辑 `config.js` 并向每台支持身份验证的服务器添加 `auth: true`。若要在服务器端启用身份验证，请参阅“身份验证”。

2.9.12 配置 Windows 执行器

https://docs.stackstorm.com/install/config/windows_runners.html

> **备注**
>
> Windows 执行器当前还是处于 beta 测试阶段，也就是可能会出现粗糙的边缘、事情可能发生断裂。
>
> 如果你遇到问题，请联系我们，我们将会尽力帮助你。

1）前置条件要求

要支持运行 Windows 执行器的 action 执行服务，该主机上需要安装以下依赖项软件：

- `smbclient` >= 4.1，命令行 Samba 客户端(Ubuntu 上的 `smbclient` 软件包 、Fedora 上的 `samba-client` 软件包)
- `winexe` >= 1.1，Windows 主机上支持远程执行命令的命令行工具

Samba 客户端在标准的 APT、Yum 仓库中就包含了。在 Ubuntu 上安装，只要执行：

```
sudo apt-get install smbclient
```

在 RHEL/CentOS 上安装，只要执行：

```
sudo yum install samba-client
```

`winexe` 没有包含在标准发行版的 RHEL/Ubuntu 软件仓库中，您需要自己进行编译，或从其他方式获得编译好的二进制软件包。

- **Ubuntu**：14.04 和 16.04 的安装指南和二进制软件包可以从这里下载；
- **RHEL/CentOS**：这些安装指南详细说明了如何构建支持 RHEL/CentOS

系统的 RPM 软件包

2）支持 Windows 软件版本

Windows 执行器已经在下面的 Windows 版本中测试过：

- Windows Server 2008
- Windows server 2012

我们用来与 Windows 主机通信的底层软件库文件也支持 Windows 的其他版本（如 2000/XP/2003/Vista），但我们没有在这些版本中测试过，因此我们无法保证它能正常工作。

3）配置 Window 服务器支持远程访问

要让 StackStorm 能够在 Windows 服务器上执行 action，您需要按照下面进行配置：

4）配置防火墙

为了让 StackStorm 能够管理您的服务器，需要配置 Windows 防火墙，允许来自运行 StackStorm 组件的服务器的流量通过。

为了安全起见，建议只允许来自 StackStorm 服务器的流量通过。如果您要让所有的 IP 都可以访问，可以运作下面命令：

```
netsh firewall set service RemoteAdmin enable
```

5）配置管理员用户帐号

StackStorm 需要有被管理的 Windows 主机的管理员帐号，能够满足在 Windows 主机上通过执行 action 完成脚本上传、执行任务。缺省是使用 `Administrator` 帐号登录到您的服务器上。

6）配置共享文件

在 Windows 脚本执行器执行之前，需要先把本地的 PowerShell 脚本上传到远程服务器。需要启用文件共享服务（SMB - Samba），并配置防火墙允许来自 StackStorm IP 地址的浏览通过，可以访问共享服务端口。

另外，您需要创建一个 StackStorm 能够上传脚本文件的共享文件夹。缺省 StackStorm 将把文件上传到共享名 `C$`。

7）配置 PowerShell

- 设置 PowerShell 允许执行脚本的执行策略，详见 <https://technet.microsoft.com/en-us/library/ee176961.aspx>
- 为了确保缺省的 `powershell.exe` 能够兼容您计划执行的脚本，可以打开 PowerShell，执行下面命令确认软件版：

```
PS C:\> $PSVersionTable
Name                           Value
----                           -----
PSVersion                      4.0
...
```

8）其他资源和链接

- 开启或关闭防火墙规则支持文件或打印机共享；
- 开启或关闭防火墙规则支持远程桌面

2.10 软件升级

https://docs.stackstorm.com/install/upgrades.html

当 StackStorm 的新版本发布时，将会发布到 APT 和 Yum 仓库中。您可以使用标准的 Linux 安装包管理工具来安装这些升级包。

作为一般升级过程的一部分，需要在重新启动 StackStorm 服务之前，您需要先运行脚本来升级 Mstral 数据库。请参阅下面的详细信息。根据要升级的版本，您可能需要运行其他迁移脚本。

如果您跳过了一个版本，并且正在升级到下一个更新的版本，请确保您也运行了跳过这个版本的迁移脚本。

1）一般升级过程

下面是标准的升级过程：

- 停止 `st2*` 服务，检查确认所有的进程都终止：

```
sudo st2ctl stop
ps auxww | grep st2
```

如果还有 *st2* 相关进程运行着，就用 *kill -9* 杀死进程。

- 使用对应系统的软件包升级工具来升级 StackStorm：

Ubuntu:

```
sudo apt-get install --only-upgrade st2 st2web st2chatops st2mistral
```

RHEL/CentOS:

```
sudo yum update st2 st2web st2chatops st2mistral
```

- 升级 Mistral 数据库

```
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf upgrade head
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf populate |
grep -v -e openstack -e keystone
```

警告

数据库升级时必须停止 mistral 和 mistral-api 服务。如果在 mistral-db-manage 执行结束之前重启数据库服务，那么升级命令 `mistral-db-manage upgrade head` 将会失败。

- 执行升级脚本，详见下面制定版本的升级脚本
- 确保所有内容都注册了

```
sudo st2ctl reload --register-all
```

- 启动 StackStorm 服务

```
sudo st2ctl start
```

2）特定版本升级脚本

我们记录了不同版本的升级说明。升级说明中记录每个版本发生的重大变更。您还可以查看每个版本更详细的变更记录。

下面章节将列出升级到相应版本时需要运行的迁移脚本。如果需要跨多个版本进行升级，请确保执行需要跳过的版本的对应脚本：

v2.5

- 如果安装了 DC Fabric Automation Suite 1.1，则必须将其升级到> = V1.1.1 的版本。按照这些指示。

v2.4

- 现在 ChatOps 使用的是 Node.js v6，以前使用的是 v4。应按照下列过程进行升级

Ubuntu:

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
sudo apt-get install --only-upgrade st2chatops
```

RHEL/CentOS:

```
curl -sL https://rpm.nodesource.com/setup_6.x | sudo -E bash -
sudo yum clean all
sudo rpm -e --nodeps npm
sudo yum upgrade st2chatops
```

要在 RHEL 或 CentOS 上的运行 Extreme Workflow Composer 的用户，必须在升级软件包后，执行下面命令

```
sudo /opt/stackstorm/st2/bin/pip install --find-links /opt/stackstorm/share/wheels --no-index --quiet --upgrade st2-enterprise-auth-backend-ldap
```

这是一个已知的问题，将会在未来的版本中解决。这仅适用于 Extreme Workflow Composer 用户。对于那些只是使用开放源码 StackStorm 的场景，这并不是必需的。

v2.2

- Mistral 的数据库结构已经发生改变了。不再使用 executions_v2 库表，该库表已经被分成 workflow_executions_v2、task_executions_v2、action_executions_v2。升级后，使用命令行(如 `mistral execution-list`)查询 Mstral 数据库将返回一个空表。executions_v2 中的记录还没有被删除。这些指令将从新表中读取。目前还没有迁移脚本将从现有 executions_v2 记录迁移到新表中。若要从 execution_v2 中读取数据，可以使用 psql 读取或在安装有 python-misalclient 旧版本的独立 python 虚拟环境中访问。

警告

请确保按照上面列出的通用步骤对数据库进行升级。

- 如果您在运行 `mistral-db-manage upgrade head` 时看到 `event_triggers_v2 already exists` 错误事件，这意味着在运行 mistral-db-manage 命令之前运行着 mstral 服务。SQLAlchemy 会在更新的数据库架构时会自动创建新表，并且会与 mistral-db-manage 命令冲突。若要

进行回退，需要打开 psql shell、手动删除新表，再重新运行 mstral-db-manage 命令。下面是从错误中恢复的示例脚本：

```
sudo service mistral-api stop
sudo service mistral stop
sudo -u postgres psql
\connect mistral
DROP TABLE event_triggers_v2;
DROP TABLE workflow_executions_v2 CASCADE;
DROP TABLE task_executions_v2;
DROP TABLE action_executions_v2;
DROP TABLE named_locks;
\q
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf upgrade head
/opt/stackstorm/mistral/bin/mistral-db-manage --config-file /etc/mistral/mistral.conf populate
sudo service mistral start
sudo service mistral-api start
```

v2.1

- Datastore 模型迁移，范围名称现在是 st2kv.system 和 st2kv.user，不再是 system 和 user。

```
/opt/stackstorm/st2/bin/st2-migrate-datastore-scopes.py
```

- 我们正在测试可插拔的执行器（见升级说明）。现在的执行器将像其他内容一样被显式注册。

```
/opt/stackstorm/st2/bin/st2-migrate-runners.sh
```

- 升级完成后，需要重启 st2ctl restart 或重载 st2ctl reload 服务，新软件包的功能才能正常工作。一些软件包管理 action 和 workflow 已经变更。

v1.5

- Datastore 模型迁移

```
/opt/stackstorm/st2/bin/st2-migrate-datastore-to-include-scope-secret.py
```

3）内容转移

在有些情况下，您可能需要自动地将 StackStorm 从一个实例转到另一个台主机或重新部署。为此，需要一个新的 StackStorm 实例，或迁移内容。考虑到“基础设施即代码”的方法，所有 StackStorm 内容和部件都是简单的文件，应该按照源代码管理方式保存。

(1) 使用软件包的安装程序，在全新的实例上安装 StackStorm Version_New

(2) 从 VERSION_OLD 旧实例中打包所有的 pack，把他们保存在某一 SCM 中，像 git（您应该已经使用很久了），每一个 pack 都必须在自己的仓库中。

(3) 保存 st2 数据库中的键值对：st2 key list -j > kv_file.json

(4) 从 SCM 中提取 pack。如果 SCM 是 git，您可以直接使用 st2 pack install <repo-url>=<pack-list>>安装。

(5) 重新配置所有外部的服务，指向新的实例

(6) 把键值加载到 datastore：st2 key load kv_file.json。如果对 1.5 之前的版本进行升级，您需要调整包括 scope 和 secret 的 JSON 文件。见迁移脚本 /opt/stackstorm/st2/bin/st2-migrate-datastore-to-include-scope-secret.py。

(7) 从 VERSION_OLD 服务器下/var/log/st2/*.audit.log 备份审计日志，并转移到安全的地方。注意，在做转移过程中就执行历史记录不可见，但转移完成后完整的审计记录可以在日志文件中找到。

2.11 软件卸载

https://docs.stackstorm.com/install/uninstall.html

在自动化场景中，我们深信并强烈建议服务器应该是按照野牛方式而不是宠物方式处理。因此，我们建议直接销毁 VM 或容器，而不是卸载 StackStorm。

不幸的是，有些用户在维护的环境中很难得到一台按需全新 VM，或者需要在 StackStorm 安装失败的基础上重新安装。这里我们提供了如何卸载 StackStorm 及其应用的操作指南。

警告

- 这里给出的指南将会删除数据；
- 如果尝试从安装失败的系统中恢复，下面一些步骤将会失败。按照过程操作，忽略任何错误；
- 如果系统上还运行其他应用，处理过程要特别小心。特别是有那些还要使用的 RabbitMQ、MongoDB、Nginx 或 PostgreSQL 等组件，您需要手工删除相关的数据库和配置文件；

- 删除 StackStorm 软件包时不会自动删除您已经安装的所有依赖软件。我们无法知道一开始时哪些已经安装了，也无法完全确保删除有些依赖软件是安全的，这些可能会余留在您系统中。

卸载过程概述

下面是卸载大致过程：

1）停止服务

2）删除软件包

3）删除 StackStorm 系统用户

4）删除数据库和其他依赖软件

5）删除仓库

6）清理残留日志、配置文件及目录

不同的 Linux 发行版过程略有不同，下面重点将说明。执行您对应的发行版的指引指令。

1）停止服务

- Ubuntu 系统

```
sudo st2ctl stop
sudo service nginx stop
sudo service postgresql stop
sudo service mongod stop
sudo service rabbitmq-server stop
```

- RHEL/CentOS 6.x

```
sudo st2ctl stop
sudo service nginx stop
sudo service postgresql-9.4 stop
sudo service mongod stop
```

- RHEL/CentOS 7.x

```
sudo st2ctl stop
sudo systemctl stop nginx
sudo systemctl stop postgresql
sudo systemctl stop mongod
sudo systemctl stop rabbitmq-server
```

2）删除软件包

- Ubuntu 系统

如果您只是使用 StackStorm：

```
sudo apt-get purge st2 st2mistral st2chatops st2web
```

如果已经安装了 Extreme Workflow Composer，就执行：

```
sudo apt-get purge st2 st2mistral st2chatops st2web bwc-ui st2flow
```

- RHEL/CentOS

如果您只是使用 StackStorm：

```
sudo yum erase st2 st2mistral st2chatops st2web st2python
```

如果已经安装了 Extreme Workflow Composer，就执行：

```
sudo yum erase st2 st2mistral st2chatops st2web st2python bwc-ui st2flow
```

3）删除 StackStorm 系统用户

- Ubuntu/RHEL/CentOS

```
sudo userdel -r stanley
sudo rm -f /etc/sudoers.d/st2
```

4）删除数据库和其他依赖软件

- Ubuntu

```
sudo apt-get purge mongodb-org* postgresql* rabbitmq-server erlang* nginx nodejs
```

- RHEL/CentOS

```
sudo yum erase mongodb-org* postgresql* rabbitmq-server erlang* nginx nodejs
```

5）删除仓库

- Ubuntu

```
sudo rm -f /etc/apt/sources.list.d/mongo* /etc/apt/sources.list.d/nginx.list
sudo rm -f /etc/apt/sources.list.d/StackStorm* /etc/apt/sources.list.d/nodesource*
```

- RHEL/CentOS

```
• sudo rm -f /etc/yum.repos.d/mongodb-org* /etc/yum.repos.d/StackStorm*
• sudo rm -f /etc/yum.repos.d/pgdg-94* /etc/yum.repos.d/nginx*
  /etc/yum.repos.d/nodesource*
```

5）清理残留文件

在协助软件包时，有些文件、目录还会存在，这一步就是最后删除这些

- Ubuntu

```
sudo rm -rf /etc/st2 /opt/stackstorm
sudo rm -rf /var/log/st2 /var/log/mistral /var/log/mongodb
```

```
sudo rm -rf /var/lib/mongodb /var/run/mongodb.pid
```

- RHEL/CentOS

```
sudo rm -rf /etc/st2 /etc/mongod* /etc/rabbitmq /etc/nginx /opt/stackstorm
sudo rm -rf /var/log/st2 /var/log/mistral /var/log/mongodb /var/log/rabbitmq /var/log/nginx
sudo rm -rf /var/lib/pgsql /var/lib/rabbitmq /var/lib/mongo
```

到这里，您系统将不能再运行任何与 StackStorm 相关的服务，所有主要依赖软件也已删除，您可以重新安装 StackStorm 或把该系统用于其他应用。

3 快速入门

https://docs.stackstorm.com/start.html

安装完 StackStorm 了吧？让我们把它运行起来。

本指南将指导您了解 StackStorm 基础知识，并帮助您构建和运行一个简单的自动化：在外部事件上触发操作的一种规则。

3.1 命令行方式探索 StackStorm

探索 StackStorm 的最佳方法是使用 CLI。从执行几条命令开始：

```
st2 --version

# Get help. It's a lot. Explore.
st2 -h

# Authenticate and export the token. Make sure ST2_AUTH_URL and
# ST2_API_URL are set correctly (http vs https, endpoint, and etc).
# Replace the username and password in the example below appropriately.
export ST2_AUTH_TOKEN=`st2 auth -t -p 'Ch@ngeMe' st2admin`

# List the actions from the 'core' pack
st2 action list --pack=core
st2 trigger list
st2 rule list

# Run a local shell command
st2 run core.local -- date -R

# See the execution results
st2 execution list
```

```
# Run a shell command on remote hosts. Requires passwordless SSH configured.
st2 run core.remote hosts='localhost' -- uname -a
```

所有对 StackStorm 的操作也可以采用 REST API、Python 和 JavaScript 绑定等方式。有关详细信息，请查看 CLI 和 Python 客户端参考。

您还可以通过 Web UI 做大量工作：查看历史记录、运行 action、配置 rule、安装 Pack……，在 https://{YOUR_ST2_IP}上查看。登录方式与通过 `st2` 命令行的登录相同。默认值是 `st2admin`/`Ch@ngeMe`。

3.2 身份认证

您几乎肯定会启用身份验证。最简单的方法是通过命令行方式登录：

```
st2 login st2admin --password 'Ch@ngeMe'
```

这将得到一个身份验证的令牌，并把它缓存。

还有其他身份验证方式可选：请查阅相关文档以获得更多详细信息。

3.3 使用 Action

StackStorm 提供了几个通用 action。可以通过从社区获取操作或使用现有脚本(稍后将更详细介绍)轻松扩展各种 action。

使用 `st2 action list` 浏览目录。Actiont 通过 `ref` 引用。这采用了 `pack.action_name` (例如 `core.local`)形式。

通过运行 `st2 action get <action>`或 `st2 run <action> --help` 来了解一个 action，这将给您详细描述、以及 action 参数。下面告诉您如何从命令行运行它，或者在 rule 和工作流中使用它。

```
# List all the actions in the library
st2 action list

# Get action metadata
st2 action get core.http

# Display action details and parameters.
st2 run core.http --help
```

使用 `st2 run <action> key=value positional arguments` 从命令行运行 action 操作。

```
# Run a local command
st2 run core.local -- uname -a

# HTTP REST call to st2 action endpoint
st2 run -j core.http url="https://docs.stackstorm.com" method="GET"
```

使用 `core.remote` action 在多个主机上通过 SSH 执行 Linux 命令。这假设为主机配置了可以无密码 SSH 访问，如配置 SSH 章节所述。

```
st2 run core.remote hosts='abc.example.com, cde.example.com' username='mysshuser' -- ls -l
```

备注

关于 `core.local` 和 `core.remote` action，我们使用--来区隔 action 参数，以确保关键字选项(如`-l` 或`-a`)能被正确传递给 action。或者， `core.local` 和 `core.remote`action 采用 `cmd` 参数来传递那些非常复杂的命令。

当使用命令行工具指定一个命令时，还需要转义所有变量，否则变量将由 shell 在本地进行替换。变量使用反斜杠(`\`)转义-例如`\$user`。

```
# Using `--` to separate arguments
st2 run core.local -- ls -al

# Equivalent using `cmd` parameter
st2 run core.local cmd="ls -al"

# Crazily complex command passed with `cmd`
st2 run core.remote hosts='localhost' cmd="for u in bob phill luke; do echo \"Logins by \$u per day:\"; grep \$u /var/log/secure | grep opened | awk '{print \$1 \"-\" \$2}' | uniq -c | sort; done;"
```

使用 `st2 execution` 命令检查 action 执行历史记录和 action 执行的详细信息：

```
# List of executions (most recent at the bottom)
st2 execution list

# Get execution by ID
st2 execution get <execution_id>

# Get only the last 5 executions
st2 execution list -n 5
```

到这里。你已经学会了运行 StackStorm 的 action。下面我们将事件、action

与 rule 结合起来使用。

3.4 定义一个 rule

StackStorm 使用 rule 在事件发生时运行 action 或工作流。事件通常由 sensor 来监视。当 sensor 捕捉到事件时，它会触发一个触发器。触发器将触发规则，规则会检查对应条件，如果条件匹配，则运行一个 action。非常简单吧，我们来看下面一个例子。

rule 实例：sample_rule_with_webhook.yaml

```
---
    name: "sample_rule_with_webhook"
    pack: "examples"
    description: "Sample rule dumping webhook payload to a file."
    enabled: true

    trigger:
        type: "core.st2.webhook"
        parameters:
            url: "sample"

    criteria:
        trigger.body.name:
            pattern: "st2"
            type: "equals"

    action:
        ref: "core.local"
        parameters:
            cmd: "echo \"{{trigger.body}}\" >> ~/st2.webhook_sample.out ; sync"
```

rule 是一个 YAML 文件，包含三个部分：触发器、条件和 action。此示例设置为对 webhook 的触发器作出反应，并将过滤条件应用到该触发器的内容中。

本例中的 webhook 用于侦听位于 `https://{host}/api/v1/webhooks/sample` 的 `sample`。当提交到这个 URL 时，将会触发该触发器。如果条件匹配(在本例中，有效负载中的值为 `st2`)，则有效负载将追加到 StackStorm 系统用户的主目录的 `st2.webhook_sample.out` 文件中。默认情况下，这是 `stanley`，因此文件位于`/home/stanley/st2.webhook_sample.out`。详细 Rule 解剖见规则 Rules 章节。

触发器有效载荷将引用`{{trigger}}`。如果触发器有效负载是一个有效的

JSON 对象，则可以解析，并且可以像`{{trigger.path.to.parameter}}`那样访问(它是 Jinja 模板的语法)。

规则中什么样的触发器可以使用呢？就像使用 action 一样，使用命令行来浏览触发器、了解触发器的工作原来、如何配置触发器以及负载结构是什么：

```
# List all available triggers
st2 trigger list

# Check details on Interval Timer trigger
st2 trigger get core.st2.IntervalTimer

# Check details on the Webhook trigger
st2 trigger get core.st2.webhook
```

3.5 部署一个 rule

可以将 StackStorm 配置为对 rule 的自动加载，或使用 API、命令行方式部署 rule：

```
# Create the rule
st2 rule create /usr/share/doc/st2/examples/rules/sample_rule_with_webhook.yaml

# List the rules
st2 rule list

# List the rules for the examples pack
st2 rule list --pack=examples

# Get the rule that was just created
st2 rule get examples.sample_rule_with_webhook
```

Rule 一旦创建后，webhook 开始在 `https://{host}/api/v1/webhooks/{url}` 侦听。启动 POST 时如果有 name=Joe，将检查该文件并查看它追加有效负载。

```
# Post to the webhook
curl -k https://localhost/api/v1/webhooks/sample -d '{"foo": "bar", "name": "st2"}' -H 'Content-Type: application/json' -H 'X-Auth-Token: put_token_here'

# Check if the action was executed (this shows the last action)
st2 execution list -n 1

# Check that the rule worked. By default, st2 runs as the stanley user.
sudo tail /home/stanley/st2.webhook_sample.out
```

```
# And for fun, same post with st2
st2 run core.http method=POST body='{"you": "too", "name": "st2"}'
url=https://localhost/api/v1/webhooks/sample headers='x-auth-token=put_token_here,content-
type=application/json' verify_ssl_cert=False

# And for even more fun, using basic authentication over https
st2 run core.http url=https://httpbin.org/basic-auth/st2/pwd username=st2 password=pwd

# Check that the rule worked. By default, st2 runs as the stanley user.
sudo tail /home/stanley/st2.webhook_sample.out
```

恭喜您，您已经创建、使用了第一个 StackStorm 的 rule！

3.6 部署实例

安装 StackStorm 时，将会在/usr/share/doc/st2/examples 目录下安装 rule、自定义 sensor、action 和 workflow 的示例。若要部署，只要将它们拷贝到 `/opt/stackstorm/packs/` 目录下，然后注册并重新加载：

```
# Copy examples to st2 content directory
sudo cp -r /usr/share/doc/st2/examples/ /opt/stackstorm/packs/

# Run setup
st2 run packs.setup_virtualenv packs=examples

# Reload stackstorm context
st2ctl reload --register-all
```

更多有关 action、sensor、rule 的内容，请查看 StackStorm Exchange。

3.7 Datastore

虽然大多数数据是根据 StackStorm 的需要进行存取的，但您可能需要存储和共享一些公共变量。使用 StackStorm 数据存储服务将值保存，在 rule 和工作流中用 `{{st2kv.system.my_parameter}}` 引用。

这将创建 `user=stanley` 密钥对：

```
# Create a new key value pair
st2 key set user stanley

# List the key value pairs in the datastore
st2 key list
```

更多关于数据存储的信息，请参阅数据存储（Datastore）章节。

下一章节内容

- 了解更多的 action、trigger、rule
  - ✓ 把您的脚本转成 StackStorm 的 action；
  - ✓ 学习如何编写自定义 action；
- 使用工作流将 action 整合到更高级别的自动化中—工作流 Workflows
- 查看 stackstorm.com 上的教程—这是一套越来越多使用 StackStorm 实现自动化的实用示例。

提问？问题？建议？都很欢迎！

- 支持论坛（Support Forum）
- Slack 社区频道：stackstorm-community.slack.com (注册 这里)
- 支持邮箱：support@stackstorm.com