

1 Используемые средства

- Ubuntu Linux 20.04
- Docker v19.03.13
- docker-compose v1.27.4

2 Описание файла docker-compose.yml

```
version: '3.2'
services:
  jenkins:
    container_name: jenkins
    build:
      context: jenkins
      dockerfile: Dockerfile-alpine
      args:
        JENKINS_VERSION: 2.235
    ports:
      - 8080:8080
      - 50000:50000
    volumes:
      - /tmp:/var/jenkins_home/secrets/
  gitea:
    container_name: gitea
    build:
      context: gitea
    ports:
      - 222:22
      - 3000:3000
  openproject:
    container_name: openproject
    build:
      context: openproject
    ports:
      - 5432:5432
      - 8090:80
```

Рис. 1: структура файла docker-compose.yml

На рис. 1 можно видеть структуру файла docker compose. На каждый из трёх компонентов приходится свой компонент **services**. Контекстом каждого компонента по сути является отдельная директория. Для gitea и openproject используются Docker файлы представленные в master ветках репозитория. Для jenkins в свою очередь используется модифицированный Dockerfile на основе alpine (модификация заключалась в изменении версии на более новую, а также удаление хеша проверки целостности SHA512, так как присутствующий хэш относился к старой версии jenkins). Необходимость наличия директивы **volumes** в описании jenkins обусловлена тем, что нам необходимо получить секретный одноразовый ключ необходи-

мый при первом запуске всей инсталляции. Все порты проброшены согласно стандартной документации на каждый продукт, так имеем:

- jenkins работает на порту 8080 (web) и 50000 (jenkins agent)
- gitea работает на порту 3000 (web) и 222 (для случаев использования протокола SSH при коммитах)
- openproject работает на порту 8090 (web) и 5432 (postgresql)

3 Скрипт резервного копирования

Рис. 2: Скрипт резервного копирования

Скрипт представлен на рис.2. По сути данный скрипт производит фильтрацию запущенных контейнеров по имени, и экспортирует запущенные экземпляры в tar архивы с датой и временем. Далее архивы можно загрузить вновь используя команду `docker load -i <archive_name.tar>`

4 Демонстрация установки



Рис. 3: Начальный экран jenkins, с запросом уникального ключа

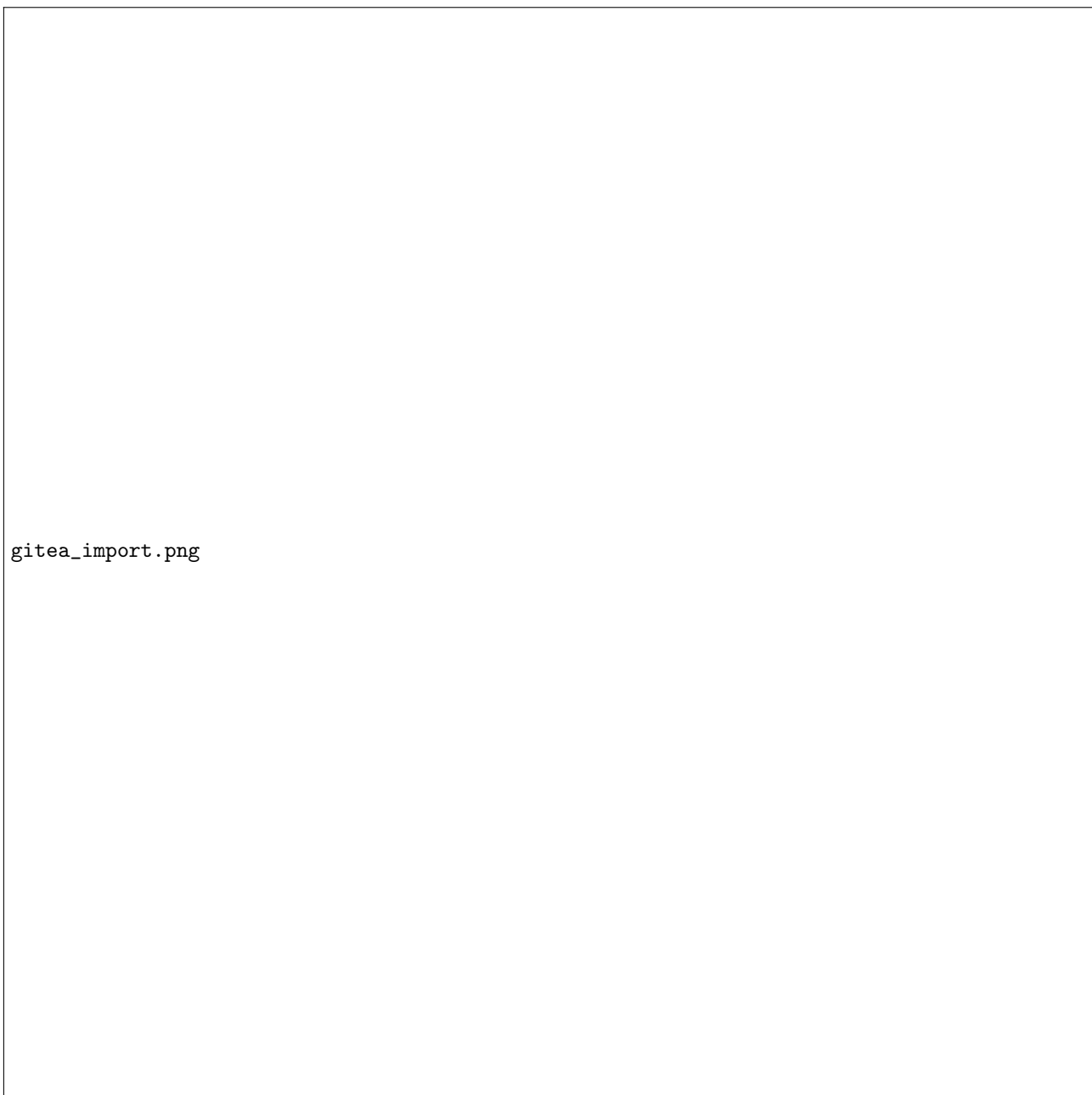


Рис. 4: Пример gitea с импортированным личным проектом

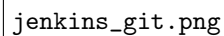
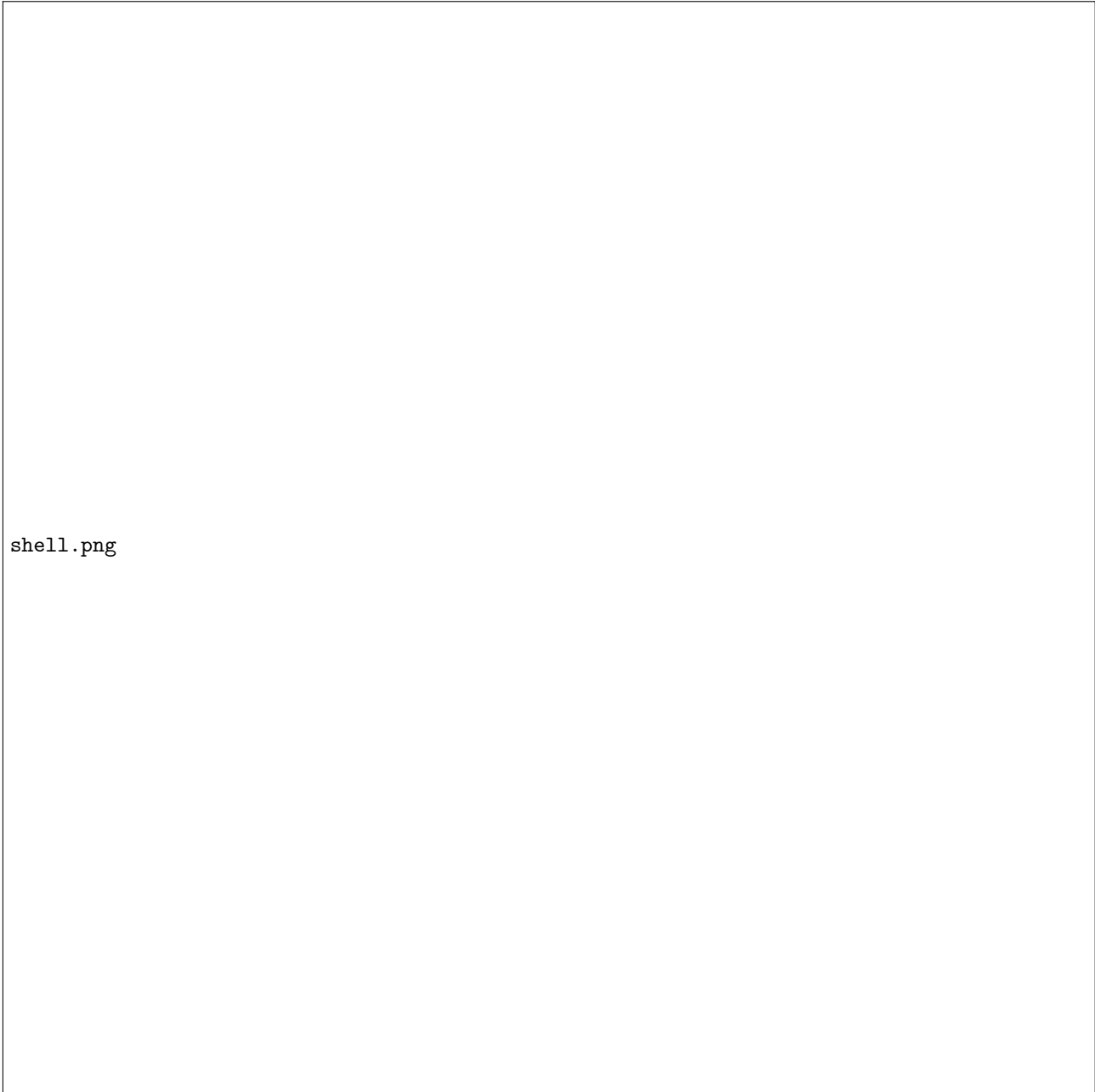
The image area is mostly blank, with the text 'jenkins_git.png' located in the lower-left corner. This likely represents a screenshot of a Jenkins configuration page for the 'git' plugin.

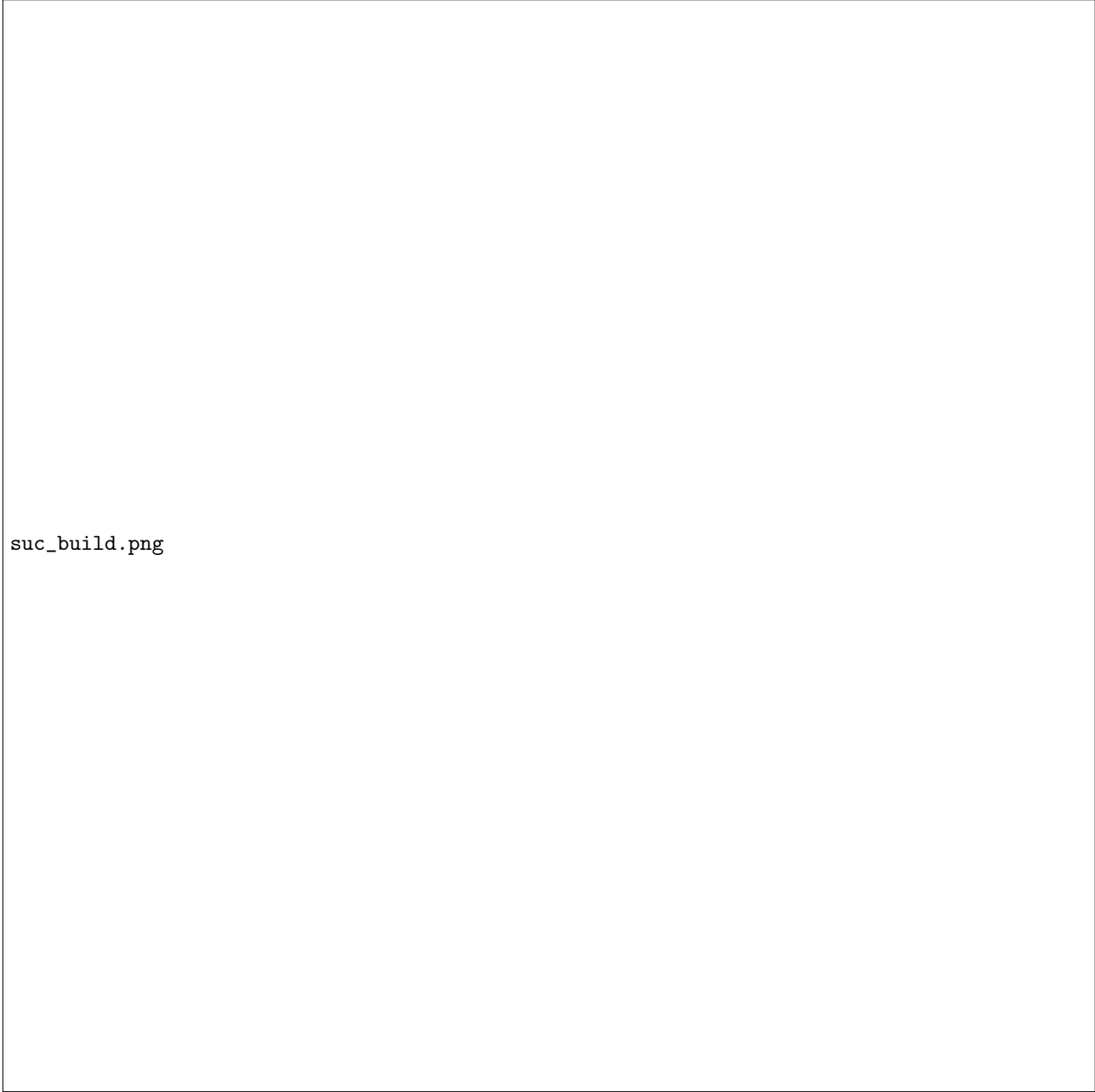
Рис. 5: Пример настройки интеграции gitea и плагина jenkins git

Для рис. 5 Стоит сделать уточнительное пояснение. Для интеграции необходимо указать полный http-адрес до репозитория. Формально инсталляция kallithea расположена по адресу `localhost:3000`. Однако данный адрес не является внутренним адресом подсети докера, для того-чтобы подключения прошло успешно необходим именно внутренний адрес. В данном случае он был выяснен путём запуска команды `docker inspect ${CONTAINER_ID} | grep IPAddr`, где `${CONTAINER_ID}` — идентификатор контейнера, в котором запущена инсталляция gitea. В качестве данных для авторизации используются данные пользователя gitea.



shell.png

Рис. 6: Пример проверочной команды для выполнения по коммиту в репозиторий



suc_build.png

Рис. 7: Пример успешной сборки проекта с помощью интеграции

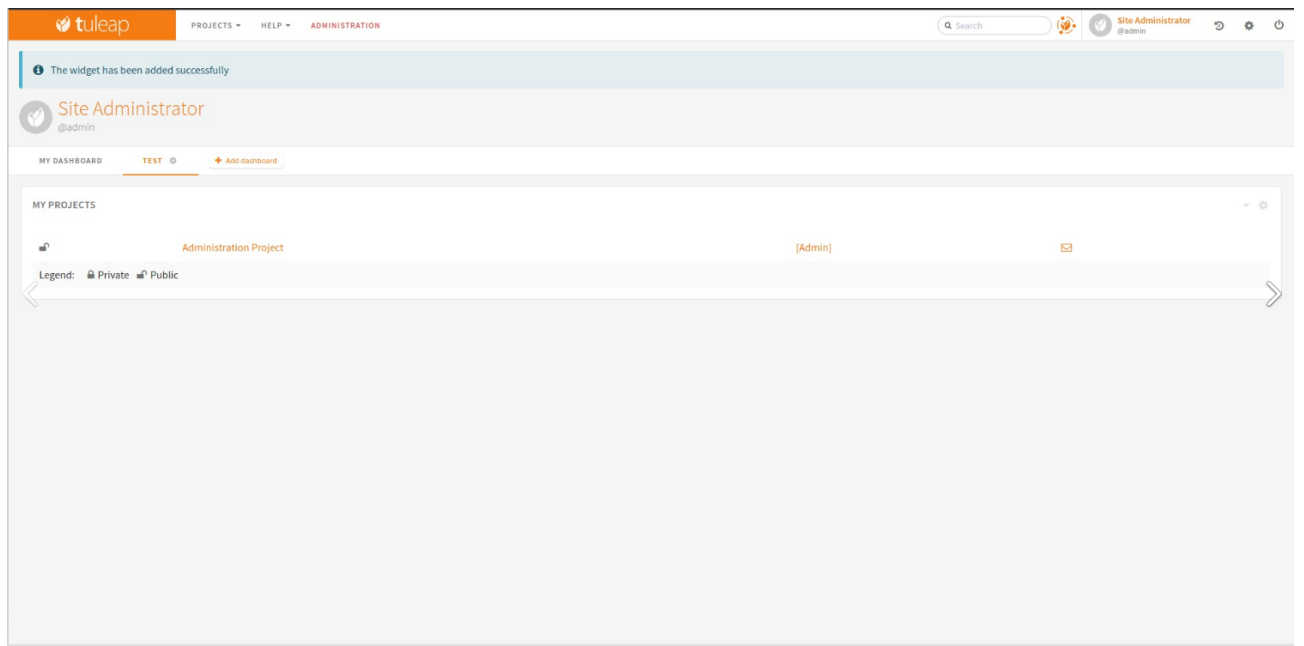


Рис. 8: Пример создания тикета в tulipe

К сожалению, согласно документации, (а также форумам поддержки всех трёх проектов), не существует интеграции kallithea с tulipe или tulipe с jenkins.