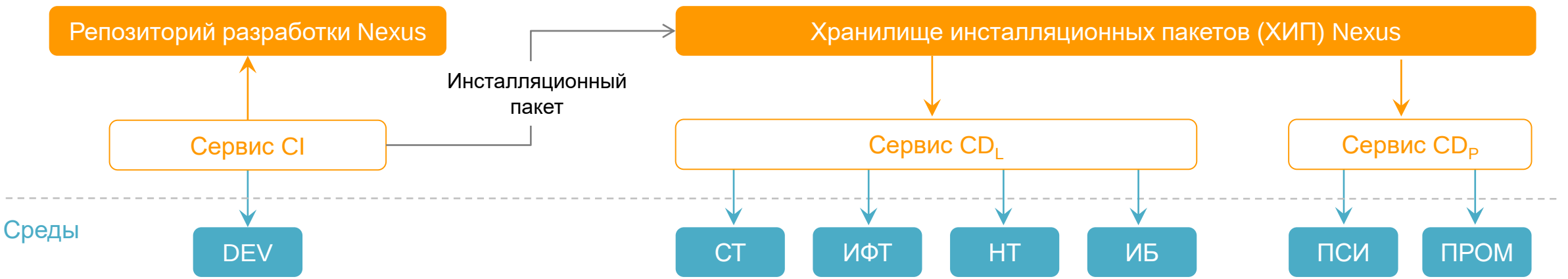
Nexus

Nexus — это централизованное хранилище Банка. Это хранилище делится на два логических элемента.

Репозиторий разработки Nexus — хранилище содержит системное ПО; различные библиотеки, использующиеся как зависимости; результаты ночных сборок; сборок из feature веток. Также этот репозиторий зеркалит различные общедоступные в Интернете репозитории, которые необходимы для процесса разработки. В общих словах этот репозиторий содержит все, что нужно для процесса **Continuous Integration**. К этому хранилищу применяются обычные требования по отказоустойчивости и по сохранности данных. Сборки в этом репозитории хранятся не более одного месяца.

Nexus Банка или ХИП Nexus— это хранилище инсталляционных пакетов (дистрибутивов) передаваемых в Банк. Этот Nexus служит гарантией того, что тот же самый дистрибутив, собранный один раз под одной версией, разворачивался и тестировался на всех этапах тестирования. То есть это хранилище служит для обеспечения непрерывной связи двух процессов — **Continuous Delivery и Continuous Deployment**. К этому хранилищу применяются повышенные требования по отказоустойчивости и по сохранности данных. То, как долго хранятся дистрибутивы в этом, репозитории определяется требованиями Банка.



Jenkins

Jenkins — это проект с открытым исходным кодом, написанный на Java, созданный решать огромное число задач по оркестрации различных процессов. В нашем случае Jenkins выбран как целевое решение для оркестрации процессов развертывания как на средах тестирования в СБТ, так и на средах приемо-сдаточных испытаний и промышленных средах в самом Банке.

Не предполагается, что **Jenkins** будет оркестрировать весь процесс от среды разработки до промышленной среды. С помощью Pipeline plugin будут оркестрированы отдельные части процесса, например, развертывание ОРД в среды СТ и в среды ИФТ – это два разных Pipeline процесса в рамках одной проектной области в сервисе CD Jenkins .

Pipeline plugin — решение, реализованное в виде плагина к **Jenkins**, позволяющее воплотить принцип «процесс развертывания как код», то есть с помощью этого плагина сам процесс развертывания представлен в виде Groovy кода. Pipeline plugin также предоставляет огромное число уже готовых модулей (функций Groovy) для реализации различных задач, а также имеет подробную документацию по каждому модулю — это обеспечит низкий порог вхождения, то есть не надо обучаться языку Groovy для того, чтобы начать создавать несложные процессы развертывания.

Используя данный плагин для всех своих развертываний мы решаем несколько задач:

- Версионирование сценария развертывания — версии Groovy кода процесса развертывания будут храниться не в самом Jenkins, а в GIT репозитории, что обеспечит надежное хранение и контроль изменений.
- Задача логично вытекающая из предыдущей — передача процесса развертывания между разными командами и частями процесса — из СБТ (разработчики для ДК) в Банк (для ЦСПС и ЦИ)
- Визуализация процесса - Pipeline plugin имеет очень удобный и понятный интерфейс отображения ранее запущенных сценариев, с возможностью быстрого доступа к логам каждого этапа.

Ansible

Ansible — программное решение с открытым кодом для удаленного управления конфигурациями, реализованное на языке Python. Ansible берет на себя всю работу по приведению удаленных серверов в необходимое состояние посредством ssh протокола, необходимо лишь описать, как достичь этого состояния с помощью сценариев playbooks, выполненных в формате yaml.

С помощью **Ansible** можно решать различные задачи, например:

- Установка/удаление СПО.
- Конфигурирование СПО.
- Создание/удаление пользователей.
- Хранение пользовательских паролей/ключей в защищенном виде (Ansible Vault).
- Развертывание кода вашего ППО.
- Запуск скриптов авто-тестирования в сторонних системах.
- Управление системой создание/удаление контейнеров/виртуальных машин.

Ansible. Преимущества

Преимущества **Ansible:**

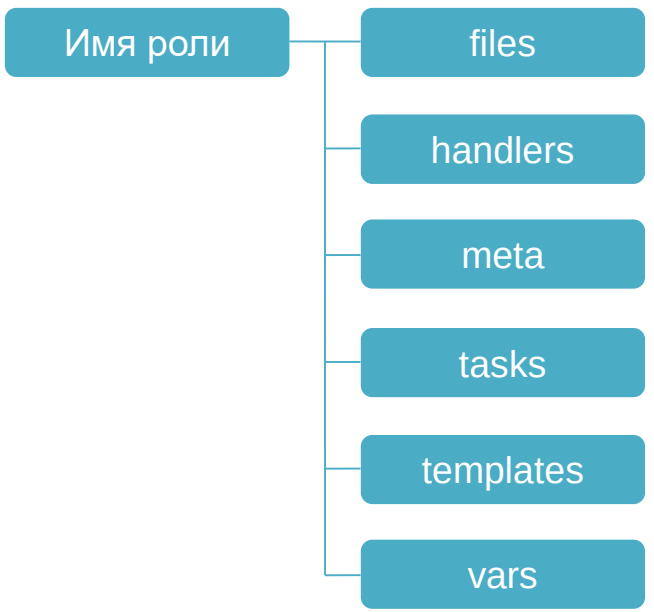
- Низкий порог вхождения. Обучиться работе с Ansible можно за очень короткие сроки.
- На управляемые узлы не нужно устанавливать никакого дополнительного ПО. Все работает через SSH.
- Код программы, написанный на Python, очень прост. Разработка дополнительных модулей не составляет особого труда.
- Декларативный язык yaml, на котором пишутся сценарии, также предельно прост.
- Встроенная возможность безопасного хранения чувствительной информации (пароли, ключи и тд) — Ansible Vault.
- Подробная и весьма просто написанная официальная документация. Она регулярно обновляется. Ее можно найти на официальном сайте.
- Ansible работает не только в режиме push, но и pull, как это делают большинство систем управления (Puppet, Chef).
- Имеется возможность последовательного обновления состояния узлов (rolling update).
- Сама структура инструмента позволяет создавать отдельные модули - роли, которые могут быть использованы разными командами в процессе поставки своего ПО. У нас имеется возможность централизованно хранить эти модули и обеспечить их использование разными командами на разных проектах.

Центральный репозиторий базовых ролей Ansible СБТ

Центральный репозиторий базовых ролей Ansible в СБТ содержит **роли Ansible**, которые централизованно разрабатываются и поддерживаются выделенными командами и должны быть использованы разными командами в процессе развертывания каждого проекта в СБТ. По смыслу этот репозиторий – это аналог официального Ansible Galaxy, но отличается тем, что в нем делятся своими наработками исключительно сотрудники СБТ и Банка.

Роль Ansible – совокупность последовательности подзадач и всех связанных с этими подзадачами данных и вспомогательных элементов в одном месте для выполнения одной функциональной задачи. По сути это логические элементы, с помощью которых выстраивается весь процесс настройки или развертывания приложения. То есть будет отдельная роль «Установка WAS», «Установка Nginx», «Установка приложения в WAS» и тд.

Обобщенная структура роли:



содержит файлы, которые будут скопированы на настраиваемые стенды. Также может содержать скрипты, которые позже будут запускаться на стендах.

обработчики, которые будут использоваться при выполнении операционных задач (таких как перезагрузка сервисов и т. д.)

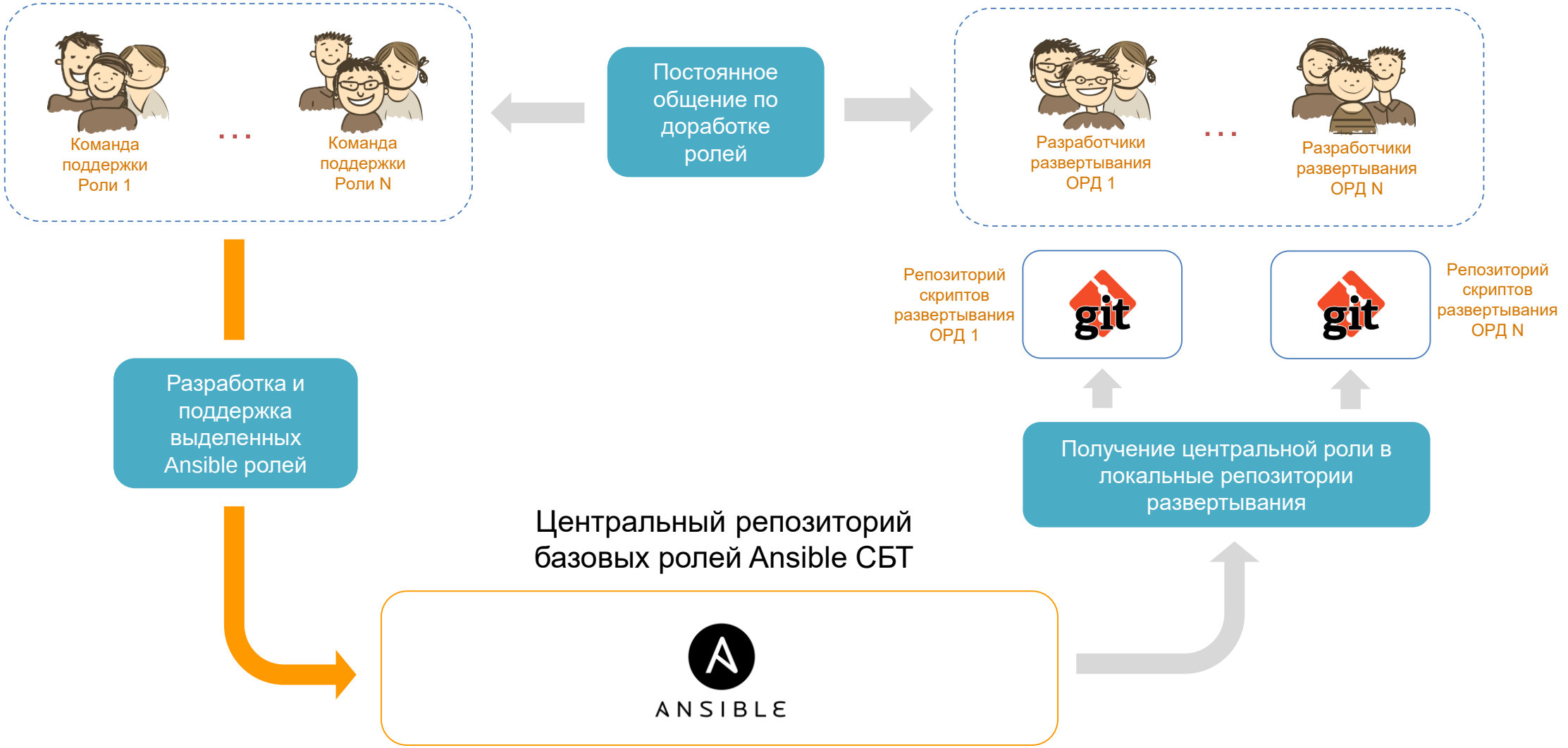
мета-данные об авторе роли, контактах для связи. Также описание зависимостей от других ролей.

это по сути основная папка роли. Здесь содержится playbook со всеми задачами, которые исполняются в рамках этой роли.

содержит файлы-темплейты (.j2) с переменными. Эти темплейты в основном используются для конфигурационных файлов.

содержит локальные переменные роли.

Центральный репозиторий базовых ролей Ansible СБТ



Центральный репозиторий базовых ролей Ansible СБТ. Вопросы

Как получить централизованную базовую роль в свой репозиторий Ansible?

Для этого необходимо создать файл requirements.yml в корне репозитория с содержанием типа:

```
- src: git@gitlab.sberbank.ru:mygroup/ansible-example.git
 scm: git
 version: "0.1"
```

Каждая роль будет находиться в отдельном репозитории, то есть для каждой роли, необходимой конкретному проекту, нужно будет сделать отдельное описание (как выше). В итоге у каждого проекта получится свой уникальный requirements.yml, который команда проекта должна будет поддерживать в актуальном виде. После того как requirements.yml подготовлен и актуализирован, на момент создания ИП (финальная часть CI процесса) нужно установить все роли в свой локальный репозиторий из центрального репозитории Ansible СБТ так:

```
ansible-galaxy install -r requirements.yml
```

затем запаковать подготовленную структуру вместе с самим приложением в ИП для дальнейшего развертывания на всех средах.

Что делать, если централизованная роль нам не подходит и нужны изменения?

В таком случае необходимо связаться с владельцем роли и запросить изменения. Если по каким-то причинам у владельца роли сейчас нет ресурсов на внесение изменений, возможен вариант, когда команда сама вносит изменения и создает pull request, который владелец роли рассматривает и одобряет/не одобряет.

Что делать, если необходимая роль вообще отсутствует в центральном репозитории Ansible СБТ ?

Каждая команда может создать свою роль при условии, что она будет добавлена в центральный репозиторий Ansible СБТ и команда возьмет на себя роль владельца. В таком случае команда берет на себя обязанности по доработке, сопровождению этой роли, а также обязуется поддерживать коммуникацию с теми командами, которые возможно захотят использовать эту роль в своем процессе развертывания.

Позволяется ли иметь свои собственные локальные роли?

Если роль очень специфична и с большой долей вероятности никогда никем не может быть переиспользована — да, но при условии, что будет обеспечен read-only доступ к вашему локальному Ansible репозиторию, для того чтобы сотрудники ЦЭПП могли исследовать ваши роли и исключать возможность дублирования функционала с центральными ролями.

Как происходит процесс развертывания?

Планируется использование процессов развертывания, основанных на плагине **Pipeline** целевого инструмента **Jenkins**.

Процесс развертывания в идеале должен содержать следующие логические шаги:

1. Получение всего необходимого для развертывания (ИП — приложение и скрипты развертывания, конфигурации)
2. Подготовка к восстановлению уже развернутого приложения (опциональный шаг, зависящий от архитектуры ФП)
3. Подготовка среды для развертывания (все ли папки созданы, права выданы, все ли скрипты на месте и готовы для развертывания)
4. Выполнение развертывания.
5. Проверки после развертывания (проверка успешности развертывания + проверка работоспособности (Smoke testing))
6. Если предыдущий шаг не успешен — восстановление приложения на предыдущую версию (опциональный шаг, зависящий от архитектуры ФП)
7. Информирование заинтересованных лиц.

Некоторые шаги опциональны и могут отключаться в зависимости от среды, на которую происходит развертывание и требований к конкретной ФП.

Например: среда CT может не требовать создания точки восстановления и возвращения на предыдущую версию.

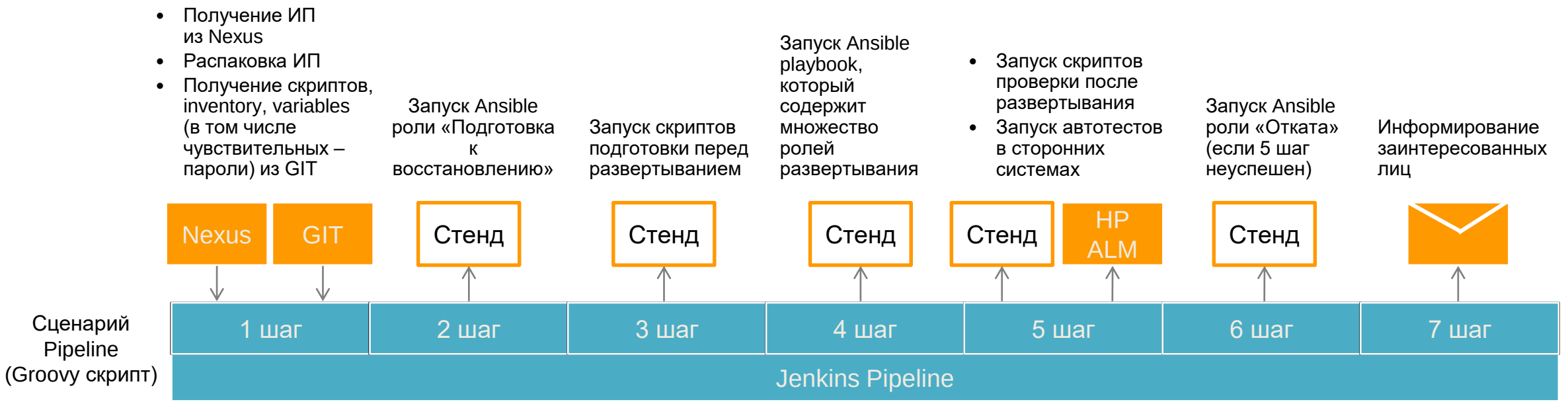
! **ВАЖНО**: это лишь логический процесс развертывания верхнего уровня, которому желательно следовать, чтобы минимизировать возможность ошибок и вреда среде, куда собственно происходит развертывание. Сами сценарии развертывания уникальны для каждого приложения.

Как реализован процесс развертывания?

В целом процесс разделен по следующим инструментам:

- **Jenkins + Pipeline plugin** в виде Groovy кода описывает общий процесс развертывания.
- **Ansible** роли – это логические элементы развертывания.
- **GIT** должен хранить все скрипты развертывания, конфигурации (сервера куда ставить – inventory, различные переменные - variables, включая чувствительные данные – пароли в зашифрованном виде)

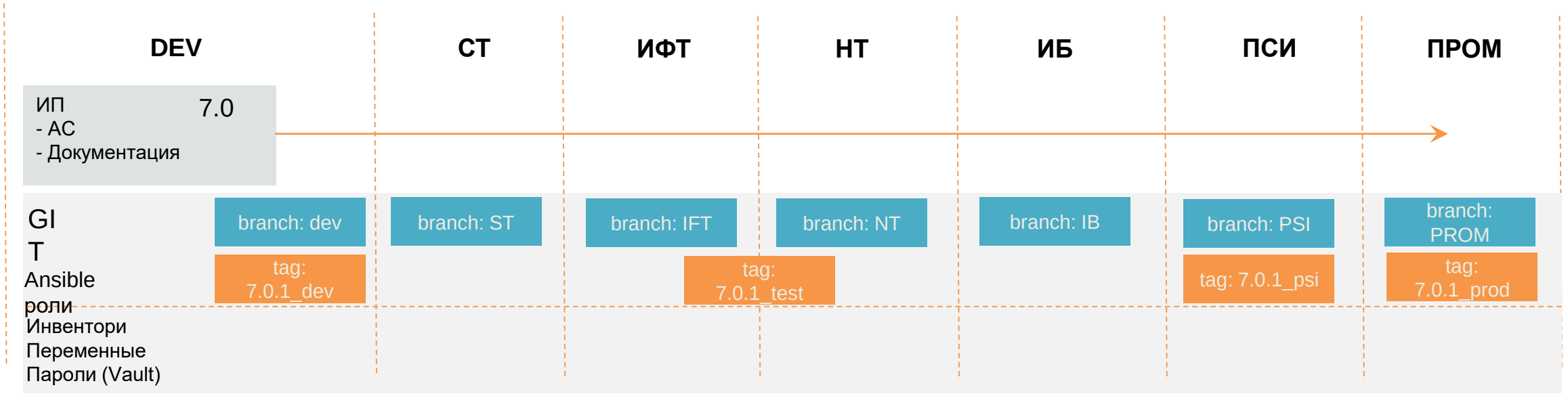
На примере процесса изложенного на слайде «Как происходит процесс развертывания?»:



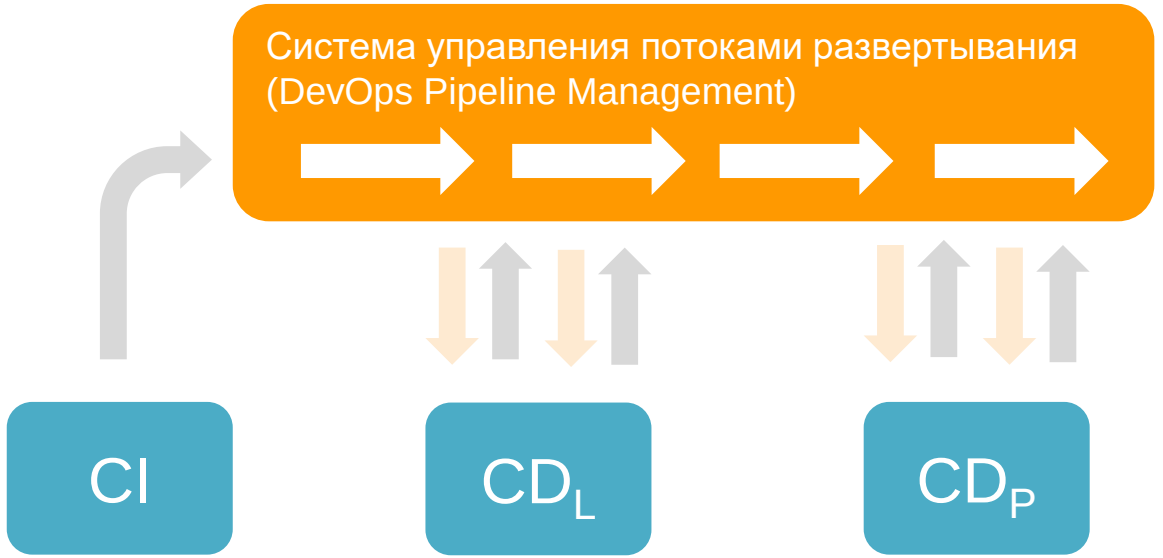
Управление конфигурациями сред

Конфигурации сред — это конфигурации, сохраняющие целевое состояние сред как на системном уровне, так и на уровне приложения. Эти конфигурации необходимы для процесса развертывания приложения.

Конфигурации сред хранятся в централизованном **GIT** репозитории. Для каждой ФП создан отдельный репозиторий с разными ветками — каждая среда развертывания имеет свою отдельную ветку и своего ответственного. За ветку «dev» отвечает команда разработки, за ветки «СТ», «ИФТ», «НТ», «ИБ» - инженер развертывания тестовых сред, за ветки «ПСИ» и «ПРОМ» - представитель отдела промышленной эксплуатации. Кроме того, чтобы зафиксировать состояние конфигураций и связать их с конкретной версией ИП, должны создаваться тэги (имя тэга — версия ИП). Таким образом, если появляются изменения в конфигурациях представители разработки создают тэг, который сообщается всем остальным командам, для того чтобы они привели свои конфигурации в соответствующее состояние.



Система управления потоками развертывания



- Управление сквозными потоками развертывания
- Визуализация сквозного маршрута инсталляционного пакета
- Реализация оперативного контроля производства технологических решений
- Сбор информации о процессах и предоставление операционной аналитики
- Сбор фактов по прохождению маршрута инсталляционным пакетом
- Синхронизация данных между сетями

Как это работает

- В конце цикла CI сформированный ИП публикуется в **Nexus** и вместе с этим в DPM отправляется сигнал о появлении новой версии ОРД.
- DPM после получения сигнала запускает поток развертывания, соответствующего ОРД.
- Поток развертывания представляет из себя набор последовательных шагов, в рамках которых DPM запускает задания в системе CD_L , затем в CD_P, получает от них статус исполнения задания и сохраняет соответствующие статусы ИП.
- DPM на специальном дашборде визуализирует процесс выполнения потока развертывания.

Интеграции процессов развертывания и тестирования

Процессы **CD_L** и **CD_P** – процессы, отвечающие за все , что касается поставки уже готового собранного приложения по всем необходимым этапам тестирования и в качестве конечной цели - в промышленную эксплуатацию. В рамках этих процессов обеспечивается гарантия целостности, неизменяемости и уникальности дистрибутива, проходящего от разработки до промышленных сред.

Однако **CD_L** и **CD_P** не дают гарантию качества и соответствия политикам безопасности Банка самого дистрибутива. Для реализации этой задачи есть отдельные процессы – Continuous Testing и тестирование информационной безопасности (ТИБ) . Эти процессы глубоко интегрированы между собой, что обеспечивает минимизацию рисков ошибок, сбоев и взломов промышленной среды и как следствие возникновения возможных нежелательных последствий для Банка.