Инженерный вестник Дона, №2 (2024)
ivdon.ru/ru/magazine/archive/n2y2024/9031

Миграция переменных сервисов с проприетарного на открытое популярное ПО

Н.Б. Лазарева

Тихоокеанский государственный университет, Хабаровск

Аннотация: В статье рассмотрен один из возможных способов переноса (миграции) переменных сервисов с проприетарного на доступное бесплатное и популярное решение, а также способы улучшения структуры и отказа от проблемных мест.
Ключевые слова: переменные, конфигурация, сервис, Octopus, Git, Vault, миграция.

В условиях современного рынка программного обеспечения не всегда удается пользоваться продуктами, разработанными за рубежом. Многие факторы, такие, как сложность/невозможность покупки/продления лицензии, санкции, а также риски, связанные с информационной безопасностью, делают сложным, а порой и невозможным использование тех или иных продуктов.

Рынок программных продуктов широк и зачастую изобилует альтернативными решениями [1], в том числе, разработанными в нашей стране. Однако встречаются программные продукты, реализующие уникальный подход к решению тех или иных задач. Это приводит к тому, что компаниям, использующим такие программные продукты, крайне сложно найти подходящую замену в случае, когда по тем или иным причинам больше нет возможности их использовать.

Так, одна из компаний столкнулась с невозможностью продления лицензии на ПО Octopus [2]. Данное ПО реализует собственный уникальный подход к реализации процессов CI/CD, а также к хранению переменных для инфраструктуры.

Подходы к реализации CI/CD - это отдельная обширная тема, которая выходит за рамки данной статьи. Здесь будет затронута тема реализации

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

хранения переменных в Octopus и возможности миграции переменных на иное ПО.

Для начала необходимо разобрать механизм работы системы хранения переменных в Octopus. В Octopus переменные можно задавать различным способом [3]. Основных способов два - на уровне конкретного проекта и в виде наборов переменных, присоединенных к различным проектам. В первом случае переменные имеют максимальный приоритет, и их область действия ограничивается только тем проектом, в котором эти переменные созданы. Во втором случае наборы переменных имеют низкий приоритет, но могут быть использованы в различных проектах. Переменные, созданные на уровне конкретного проекта, перезаписывают значения аналогичных переменных (если таковые имеются) из набора переменных, присоединенных к этому проекту.

Любая переменная может иметь различный тип значения. Основных типов два – текст и секрет. В случае типа «текст» значение задается явно и всегда отображается в интерфейсе Octopus. Значения переменных типа «секрет» никогда не отображаются в интерфейсе и обычно используются для хранения паролей, токенов и т.п.

Каждая переменная может иметь несколько значений. Это удобно, когда для различных сценариев нужно иметь различные значения одной и той же переменной. Рассмотрим на конкретном примере работу данного механизма. Предположим, что в приложении существует подсистема логирования, которая в процессе работы приложения логирует те или иные события, происходящие в приложении. Обычно в таких случаях есть возможность регулировать уровень логирования – от самого подробного, когда логируются все возможные события до самого основного, когда логируются только самые важные события. Самый подробный уровень логирования бывает необходим на этапе тестирования приложения, когда

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

нужно четко понимать, что происходит в приложении в процессе его работы. Однако, держать уровень логирования включенным на самом подробном уровне не оптимально, так как это порождает большое количество информации и существенно загружает системы сбора и хранения логов. Поэтому уровень логирования оптимально устанавливать так, чтобы логировались только основные события, и включать подробный уровень только тогда и в тех местах, когда это необходимо. Так, переменная, хранящая в качестве значение уровень логирования приложения, может иметь основной уровень по умолчанию и подробный только для конкретных шагов/окружений и т.п. Octopus предоставляет различные варианты условий, при которых будут выбираться те или иные значения одной и той же переменной в различных ситуациях (рис.1).

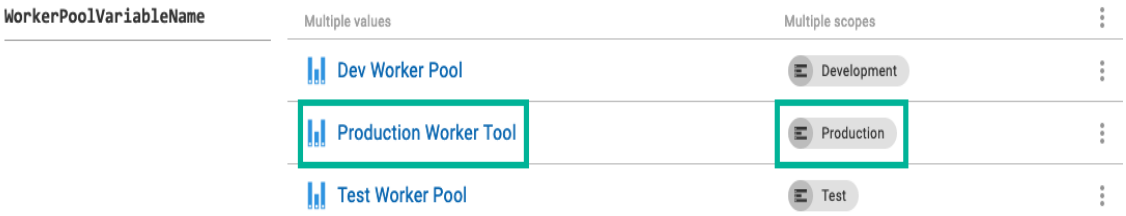

Рис. 1. – Различные значения одной переменной в Octopus

Рассмотрим преимущества и недостатки данного подхода к организации хранения переменных в Octopus. Преимущества:

1. Переменные можно объединять и наборы и использовать в большом количестве проектов, что приводит к уменьшению дублирования переменных.

2. Наборы переменных, подключенные к разным проектам, позволяют централизованно менять значения переменных сразу для нескольких проектов.

3. Система приоритетов переменных позволяет выстроить иерархию

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

значений и перезаписывать определенные переменные при необходимости.

4. Возможность задания сразу нескольких значений переменной упрощает работу с переменными при большой вариативности значений.

5. Значения переменной могут содержать в себе значения других переменных, что позволяет конструировать значения переменной, исходя из определенных условий.

Все преимущества также являются и недостатками:

1. Наборы переменных могут быть избыточными для тех или иных проектов, когда для них нужно только определенное подмножество переменных их набора.

2. Нет возможности контролировать фактическое использование переменных из набора в проектах, что приводит к появлению ненужных, устаревших переменных и/или их значений. Удаление переменных, а также добавление/изменение/удаление значений может привести к ошибкам в тех или иных проектах.

3. При большом количестве проектов, переменных уровня проектов и наборов переменных существенно усложняется процесс добавления/удаления переменных, так как нет возможности просмотреть все существующие переменные и их значения в одном месте и выполнить сквозной поиск.

4. Система приоритетов переменных не позволяет проверить корректность конфигурации до момента запуска определенных шагов проекта, в котором эти переменные используются.

5. Подстановка значений переменных в значения других переменных существенно усложняется, если уровень вложенности таких подстановок превышает два уровня, а также в случаях, когда значения переменных объединяют в себе значения нескольких других переменных.

6. Отсутствие удобного механизма аудита, системы контроля внесения правок в переменные, а также способов возврата к предыдущим версиям

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

переменных, что существенно влияет на надежность и стабильность инфраструктуры, использующей переменные.

С учетом выявленных недостатков, мигрировать переменные из Octopus в другое программное решение стоит не напрямую, а с определенными изменениями.

Основная идея состоит в том, чтобы поместить все переменные под управление системой контроля версий Git [4]. Это сразу устранит проблемы с версионированием, контролем изменений и аудитом, тем самым упразднятся основные недостатки текущего подхода.

Система Git сама по себе не реализует принципов приоретизации переменных, наборов переменных, подстановок значений и прочую логику. Для реализации этой функциональности понадобится написать собственные скрипты, автоматизирующие данные операции. Поскольку процессы CI/CD также должны быть перенесены из Octopus в другое программное обеспечение, написание или изменение скриптов, используемых в этих процессах, будет неизбежным. В этих условиях написание скриптов для обработки переменных и подстановка результатов в процесс CI/CD не является существенной трудозатратой.

С учетом необходимости написания скриптов есть смысл реализовать те или иные функции и логику для работы с переменными таким образом, чтобы избежать остальных недостатков текущего подхода. Для этого подготовим следующую структуру каталогов в репозитории системы контроля версий Git (рис.2).

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

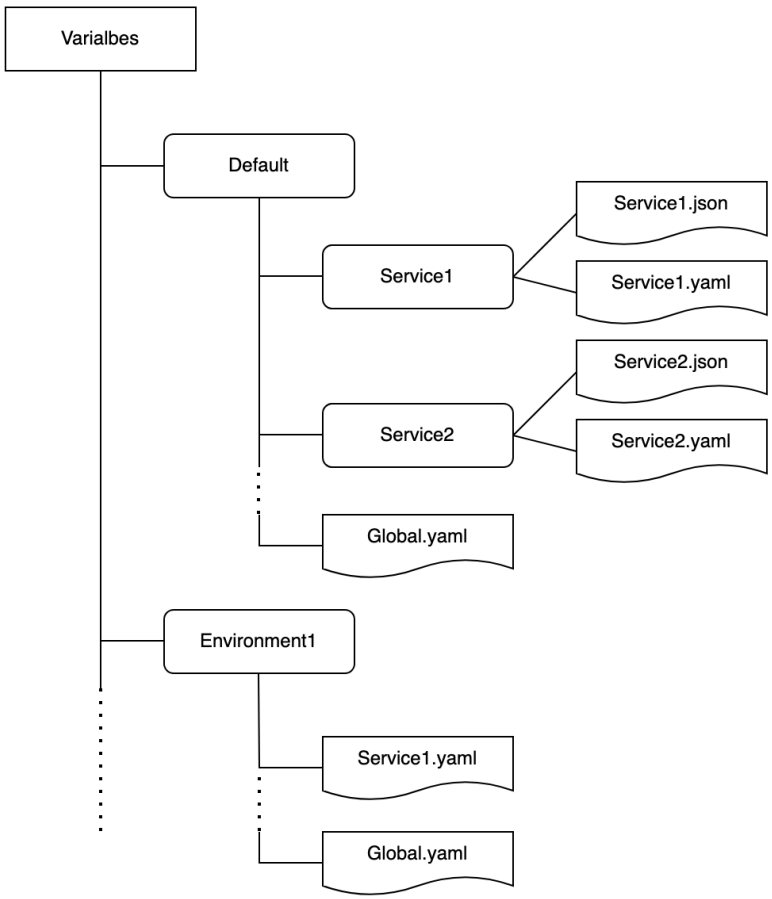

Рис. 2. – Структура репозитория с переменными

Каталог Default - это источник значений по умолчанию для каждого сервиса. Так, для сервиса Service1 файл Service1.json является хранилищем структуры данных, которую ожидает сервис в качестве конфигурации. Этот файл частично или полностью шаблонизирован. Если значение переменной статично и ни при каких условиях не изменяется, оно задается в структуре json явно (рис.3). Если переменная может иметь несколько значений, то вместо конкретного значения в json вставляется заглушка-переменная. Источников данных для этой заглушки может быть несколько. Основной источник – файл Service1.yaml. В данном файле содержатся значения по умолчанию для переменных-заглушек в json-файле. Для каждого окружения можно добавлять другие значения переменной. Это реализуется созданием файла Service1.yaml в каталоге окружения, например, Environment1. В таком

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

случае, значение переменной, заданной на этом уровне, является более приоритетным по сравнению со значением, заданным в каталоге Default/Service1 (рис.4).

```
{
    "variable1": "value1",
    "variable2": "{{ variable2 }}",
    "block1": {
        "variable3": "{{ variable3 }}"
    }
}
```

Рис. 3 – json-файл для Service1

```
---
variable2: value2
variable3: value3
```

Рис. 4 – Пример json-файла

Также можно задавать значения переменных в Global.yaml-файлах в директории Default и директориях окружений. Это может быть полезно в случае, когда значения переменных могут или должны использоваться сразу в нескольких сервисах. В таком случае можно добиться централизованного управления значениями таких переменных и избежать дублирования.

Скрипты автоматизации, запускаемые в процессе установки сервиса, берут за основу json-файл [5] и накладывают на него значения, заданные в yaml-файлах на всех уровнях в зависимости от того, на какое окружение происходит установка сервиса.

Проанализируем, всех ли недостатков удалось избежать в такой схеме хранения данных:

1. Наборы переменных больше не присутствуют как сущности, нет проблем с ними связанных.

2. Есть возможность контролировать использование переменных, в том числе, общих для нескольких сервисов. Этому способствуют json-файлы, которые несут однозначные структуры данных, ожидаемые сервисами.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

Опираясь на них, можно понять, какие переменные устарели, а какие используются.

3. Также важен тот факт, что репозиторий хранит переменные для всех сервисов и можно пользоваться сквозным поиском по всему репозиторию для поиска переменных и их значений.

4. С помощью скриптов можно легко реализовать просмотр итогового набора переменных для каждого сервиса под каждое окружение без необходимости запускать шаги установки.

5. Подстановка переменных в значения других переменных технически присутствует в данной структуре, однако ее глубина сильно сокращается, так как упразднены наборы переменных.

6. Благодаря Git появились механизмы аудита [6], контроля версий [7]. Более развитое ПО, работающее на основе Git (GitLab [8], GitHub), позволяет использовать процессы согласования изменений в переменных и добавить механизмы синтаксической и логической проверки всех файлов.

7. Дополнительно, для хранения чувствительных (секретных) переменных можно использовать специализированное ПО (например, Vault [9]). Внутри Vault можно повторить аналогичную структуру каталогов и файлов, чтобы упростить понимание наследования значений. Адаптация скриптов для добавления значений секретных переменных в итоговую конфигурацию сервиса тривиальна. В Vault реализована собственная система версионирования и аудита, которыми удобно пользоваться при работе с секретами.

Таким образом можно перенести структуру хранения переменных с проприетарного решения на общедоступные аналоги. Полученная в таком случае гибкость позволяет реализовать весь необходимый функционал и избежать проблемных мест. Сложность скриптов автоматизации в данном

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

случае невысока и позволяет довольно быстро реализовать те или иные сценарии работы с переменными.

Структура переменных не привязана к конкретной реализации инфраструктуры. Сервисы могут устанавливаться на различные площадки (например, Kubernetes [10], виртуальные машины, bare-metal и т.д.) и в любом случае может быть использован такой подход к хранению переменных.

Литература

1. Лазарева Н. Б. Оптимальный подход к разработке программного обеспечения с использованием современных методологий и технических средств // Инженерный вестник Дона, 2020, №10. URL: ivdon.ru/ru/magazine/archive/n10y2020/6625

2. Steve Fenton. Exploring Octopus Deploy. – Lulu.com, 2015. – P. 166.

3. Gaurav Agarwal. Modern DevOps Practices. – Packt Publishing Ltd, 2021. – P. 530.

4. Тармосина А.С., Пчелинцев А.Н. Внедрение системы контроля версий GIT – компании // Синергия наук, 2017, №18. URL: synergy-journal.ru/archive/article1497

5. Ben Smith. Beginning JSON. – Apress, 2015. – P. 339.

6. Верещагина Е.А., Рудниченко А.К., Колесникова Д.С. Windows Management Instrumentation как способ мониторинга и аудита ИТ-инфраструктуры предприятия // Инженерный вестник Дона, 2019, №8. URL: ivdon.ru/ru/magazine/archive/n8y2019/6125

7. Aplaev G. Software Testing Automation Tips. – Springer, 2017. – P. 68.

8. Чакон С., Штрауб Б. Git для профессионального программиста. – Питер, 2022**.** — 496 с.

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024

9. Vault Documentation. URL: developer.hashicorp.com/vault/docs?product_intent=vault (дата обращения: 20/01/2024).

10. Burns B., Beda J., Hightower K., Lachlan Evenson L. Kubernetes: Up and Running: Dive into the Future of Infrastructure. – O'Reilly Media, 2022. – P. 354.

References

1. Lazareva N.B. Inzhenernyj vestnik Dona, 2020, №10. URL: ivdon.ru/ru/magazine/archive/n10y2020/6625

2. Steve Fenton. Exploring Octopus Deploy. Lulu.com, 2015. P.166.

3. Gaurav Agarwal. Modern DevOps Practices. Packt Publishing Ltd, 2021. P. 530.

4. Tarmosina A.S., Pchelincev A.N. Sinergiya nauk, 2017, №18. URL: synergy-journal.ru/archive/article1497

5. Ben Smith. Beginning JSON. Apress, 2015. P. 339.

6. Vereshhagina E. A., Rudnichenko A. K., Kolesnikova D. S. Inzhenernyj vestnik Dona, 2019, №8. URL: ivdon.ru/ru/magazine/archive/n8y2019/6125

7. Aplaev G. Software Testing Automation Tips. Springer, 2017. P. 68.

8. Chakon S., Shtraub B. Git dlja professional'nogo programmista [Git for the professional programmer]. Piter, 2022. 496 p.

9. Vault Documentation. URL:developer.hashicorp.com/vault/docs?product_ intent=vault (accessed 20/01/2024).

10. Burns B., Beda J., Hightower K., Lachlan Evenson L. Kubernetes: Up and Running: Dive into the Future of Infrastructure. O'Reilly Media, 2022. P. 354.

Дата поступления: 7.01.2024

Дата публикации: 17.02.2024

© Электронный научный журнал «Инженерный вестник Дона», 2007–2024