La aplicación de los doce factores

X. Paridad de desarrollo / producción

Mantenga el desarrollo, la puesta en escena y la producción lo más similar posible

Históricamente, ha habido brechas sustanciales entre el desarrollo (un desarrollador que realiza ediciones en vivo en una implementación local de la aplicación) y la producción (una implementación en ejecución de la aplicación a la que acceden los usuarios finales). Estas brechas se manifiestan en tres áreas:

- **La brecha de tiempo** : un desarrollador puede trabajar en código que demore días, semanas o incluso meses en entrar en producción.
- **La brecha de personal** : los desarrolladores escriben código, los ingenieros de operaciones lo implementan.
- **La brecha de herramientas** : los desarrolladores pueden estar usando una pila como Nginx, SQLite y OS X, mientras que la implementación de producción usa Apache, MySQL y Linux.

La aplicación de doce factores está diseñada para un despliegue continuo al mantener pequeña la brecha entre el desarrollo y la producción. Mirando las tres brechas descritas anteriormente:

- Reduzca el intervalo de tiempo: un desarrollador puede escribir código y desplegarlo horas o incluso minutos después.
- Reduzca la brecha de personal: los desarrolladores que escribieron el código están muy involucrados en implementarlo y observar su comportamiento en la producción.
- Reduzca la brecha de herramientas: mantenga el desarrollo y la producción lo más similar posible.

Resumiendo lo anterior en una tabla:

	Aplicación tradicional	Aplicación de doce factores
Tiempo entre implementaciones	Semanas	Horas
Autores de código versus implementadores de código	Gente diferente	Misma gente
Dev vs entornos de producción	Divergente	Lo más similar posible

Los servicios de respaldo , como la base de datos de la aplicación, el sistema de colas o el caché, es un área donde la paridad de desarrollo / producción es importante. Muchos idiomas ofrecen bibliotecas que simplifican el acceso al servicio de respaldo, incluidos los *adaptadores* para diferentes tipos de servicios. Algunos ejemplos están en la tabla a continuación.

Tipo	Idioma	Biblioteca	Adaptadores
Base de datos	Ruby / Rails	ActiveRecord	MySQL, PostgreSQL, SQLite
Cola	Python / Django	Apio	RabbitMQ, Beanstalkd, Redis
Cache	Ruby / Rails	ActiveSupport :: Caché	Memoria, sistema de archivos, Memcached

Los desarrolladores a veces encuentran un gran atractivo en el uso de un servicio de respaldo ligero en sus entornos locales, mientras que se utilizará un servicio de respaldo más serio y robusto en la

producción. Por ejemplo, usando SQLite localmente y PostgreSQL en producción; o memoria de proceso local para el almacenamiento en caché en desarrollo y Memcached en producción.

El desarrollador de doce factores resiste el impulso de usar diferentes servicios de respaldo entre el desarrollo y la producción , incluso cuando los adaptadores teóricamente abstraen cualquier diferencia en los servicios de respaldo. Las diferencias entre los servicios de respaldo significan que surgen pequeñas incompatibilidades, lo que hace que el código que funcionó y pasó las pruebas en el desarrollo o la puesta en escena falle en la producción. Estos tipos de errores crean fricciones que desincentivan el despliegue continuo. El costo de esta fricción y la amortiguación posterior del despliegue continuo es extremadamente alto cuando se considera en conjunto durante la vida útil de una aplicación.

Los servicios locales ligeros son menos atractivos de lo que alguna vez fueron. Los servicios de respaldo modernos como Memcached, PostgreSQL y RabbitMQ no son difíciles de instalar y ejecutar gracias a los sistemas de empaque modernos, como Homebrew y apt-get . Alternativamente, las herramientas de aprovisionamiento declarativas como Chef y Puppet combinadas con entornos virtuales livianos como Docker y Vagrant permiten a los desarrolladores ejecutar entornos locales que se aproximan mucho a los entornos de producción. El costo de instalar y usar estos sistemas es bajo en comparación con el beneficio de la paridad de desarrollo / producción y la implementación continua.

Los adaptadores a diferentes servicios de respaldo siguen siendo útiles, ya que hacen que la transferencia a nuevos servicios de respaldo sea relativamente sencilla. Pero todas las implementaciones de la aplicación (entornos de desarrollador, preparación, producción) deben usar el mismo tipo y versión de cada uno de los servicios de respaldo.

한국어 (ko) | 简体 中文 (zh_cn) | Français (fr) | ภาษา ไทย (th) | Italiano (it) | Español (es) | Polski (pl) | Portugués brasileño (pt_br) | Ελληνικά (el) | 日本語 (ja) | Slovensky (sk) | Українська (reino unido) | Turco (tr) | Inglés (en) | Deutsch (de) | Русский (ru)
«Anterior
Próximo "
한국어 (ko) | 简体 中文 (zh_cn) | Français (fr) | ภาษา ไทย (th) | Italiano (it) | Español (es) | Polski (pl) | Portugués brasileño (pt_br) | Ελληνικά (el) | 日本語 (ja) | Slovensky (sk) | Українська (reino unido) | Turco (tr) | Inglés (en) | Deutsch (de) | Русский (ru)
Escrito por Adam Wiggins
Última actualización 2017
Código fuente
Descargar ePub Book
Política de privacidad