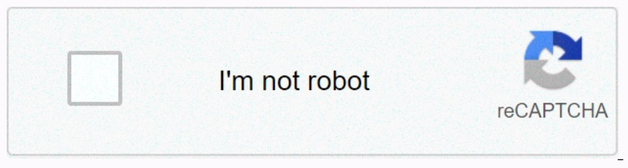


Continue

# Devops infrastructure as code terraform

Senza guardare indietro, è chiaro che la trasformazione digitale che stiamo vivendo, soprattutto nelle aree cloud, è senza precedenti. È sempre più importante, infatti, stare al passo con i tempi, adottare metodi di sviluppo innovativi ed essere pronti a soddisfare tutte le richieste che provengono dagli utenti del nostro software che stanno diventando sempre più esigenti. Che si tratti di startup o aziende, la trasformazione digitale e il cloud computing in generale hanno un grande impatto su come costruiamo il software, su come lo distribuiamo, su come lo voleremo disponibile a livello globale, su come lo volemo scalabile e accessibile a milioni di persone. A pensarci, soprattutto nel campo aziendale, richiedere un server aggiuntivo per distribuire una nuova applicazione web o evidenziare nuove macchine virtuali è stata un'operazione che ha richiesto potenzialmente settimane: siamo passati da una richiesta iniziale alla definizione dei requisiti, passando attraverso una revisione del budget, approvando manager di vario tipo per arrivare, solo giorni (o peggio settimane) più tardi per acquistare le attrezzature necessarie per la preparazione e la configurazione effettiva dell'infrastruttura necessaria. Quello che qualche anno fa era fantascienza oggi è standard: per fornire un nuovo server o database dobbiamo solo fare un paio di clic in Azure (o qualsiasi altra piattaforma) e in pochi secondi (o peggio minuti) abbiamo tutto ciò di cui abbiamo bisogno disponibile ovunque nel mondo e con la protezione dei dati. Un'infrastruttura come la formazione rapida delle risorse di CodeEffete attraverso l'interfaccia utente di un provider di cloud ha indubbiamente portato molti vantaggi al nostro lavoro. Ad esempio: non dobbiamo più aspettare il tempo necessario per acquistare le apparecchiature in modo da poter costruire rapidamente l'infrastruttura di cui abbiamo bisogno senza passare da fornitori specializzati (IBM, Dell, ecc.); Non dovremmo assumere personale specializzato nell'installazione di attrezzature, ma piuttosto possiamo riciclare le competenze DevOps; Possiamo fare a meno di acquistare un data center al premio e spendere milioni di euro per la loro configurazione e manutenzione (gestione, configurazione, backup...); Non dobbiamo buttare via le procedure che abbiamo definito con le conoscenze precedenti; L'elevata affidabilità (con protezione dei dati) e l'accessibilità sono spesso garantite in conformità con determinati standard e best practice. Per quanto sia facile andare nel cloud e creare risorse con un semplice clic, il processo è ancora manuale e, come tale, ancora potenzialmente soggetto a errori umani. Può anche essere difficile apprendere l'intera infrastruttura di realtà grande come un'azienda e capire come ridimensionare le istanze di un determinato servizio mantenendo la stessa configurazione di quelle precedenti già distribuite, spesso la documentazione non viene trovata e così via. I vantaggi del cloud naturalmente, lo sviluppo, ma anche introdurre nuovi problemi che dobbiamo risolvere. Infrastructure as Code (IaC) nasce per cercare di risolvere tutti questi problemi. L'infrastruttura come codice è un approccio alla gestione dell'infrastruttura IT in un'era di cloud, microservizi e distribuzione continua. - Keith Morris, Head of Continuous Delivery presso ThoughtWorks Europe. Mantenere l'infrastruttura come codice significa, come dice giustamente la citazione, essere più flessibili nella preparazione e nella gestione dell'intera infrastruttura. Questo può essere fatto seguendo un approccio semplice ma funzionale: l'intera infrastruttura è descritta e definita proprio come il software, e quindi può prendere tutti i vantaggi, tra cui: Mantenere lo stesso ciclo di vita: modifiche dell'infrastruttura a seconda del software, ad esempio, quando ci si sposta tra .FRAMEWORK e .NET Core potrei volere server Linux Windows invece di risparmiare sui costi e avere prestazioni migliori; Automatizzare l'intero processo di preparazione e de-preparazione, con configurazione e minimizzazione dei rischi: utile in scenari non multi-tenant in cui è necessario riprodurre la stessa infrastruttura e configurazione su N-clienti, e quindi eliminarla quando non ne abbiamo più bisogno, o quando vogliamo riprodurre con precisione l'ambiente di produzione; Velocità e facilità di preparazione: non hai bisogno di ulteriori conoscenze di sistema o persone dedicate, ma essendo tutti definiti dal codice anche gli sviluppatori possono organizzare la configurazione a loro piacere e ridimensionarla in base alle esigenze. Essendo tutto compilato come uno scenario, la distribuzione richiederà lo stesso tempo necessario per la distribuzione direttamente al portale di Azure (o ad altri fornitori) se la configurazione (manuale o automatica) ;D ocumentation: Il codice se scritto bene è autode descritto, ma nel caso in cui non sia documentazione può essere definito insieme all'infrastruttura, in modo che non sia centralizzato nel reparto IT, non nella testa di una persona; Verifica e convalida: seguendo lo stesso ciclo di vita del software (ALM), gli stessi meccanismi di sicurezza e verifica possono essere applicati tramite una richiesta di stretching, revisione del codice e pre-test; Versione: essendo definita come codice, l'infrastruttura può anche seguire git o qualsiasi altro sistema di gestione del codice sorgente e pertanto può avere un proprio numero di versione, in modo da poter eseguire il rollback, ad esempio, per un backlog delle distribuzioni; Distribuzione come modello: potenzialmente tra un'infrastruttura e un altro solo per modificare le impostazioni di input, ma le basi architettoniche sono le stesse, quindi può essere conveniente creare e riutilizzare i modelli (inclusi i modelli di terze parti); Riduzione dei costi: meno intuitivo, ma qualsiasi sistema IaC può essere integrato in un sistema di controllo dei costi (ad esempio, Prezzi di Azure), che può quindi eseguire la prima analisi cambiare e fare un controllo con la politica dell'azienda per mantenere bassi i costi. In alternativa all'analisi preventiva, possiamo fare analisi retrospettive ed eliminare facilmente le risorse che non usiamo più perché non sono più critiche. I vantaggi, come abbiamo visto, sono innumerevoli e non si fermano qui, ma per loro vi invitiamo ad approfondire l'argomento, perché i vantaggi dipendono anche dalla posizione di destinazione per la distribuzione. Come ci avviciniamo al tema dell'infrastruttura come Codice? Anche se questo può sembrare strano, come abbiamo visto più volte per DevOps, la parte più difficile è sempre cambiare la mentalità per avvicinarsi a questa nuova modalità di funzionamento, mentre la parte tecnica è, al contrario, la più semplice. Come parte dell'articolo, naturalmente proveremo a dedurre la componente tecnica, nella speranza che possa aiutare a capire perché è conveniente utilizzare meccanismi di questo tipo, e speriamo che il pensiero venga di conseguenza. Se guardiamo l'intero panorama degli strumenti disponibili, spaziamo tra chef, bambole, Ansible, CloudFormation e molti altri. Se siamo per il resto del panorama Microsoft, ci sono alcuni strumenti che ci permettono di definire l'infrastruttura come codice in Azure: ARM è la definizione di infrastruttura sotto forma di file JSON creati da Microsoft e utilizzati per distribuire le risorse tramite un gestore di risorse; Bicep, sempre creato da Microsoft e attualmente rilasciato in anteprima (e non ancora completato nella sintassi), che fissa l'obiettivo di rimuovere la complessità delle mani con una definizione più semplice. Bicep è ancora in fase di definizione della lingua, quindi è molto difficile essere in grado di parlarne ora. Tuttavia, c'è Terraform, che è esattamente a metà strada tra gli strumenti generalisti e Microsoft, che utilizza un linguaggio personalizzato chiamato HCL, adatto a qualsiasi esodi e il cui provider azure è sviluppato da Microsoft. 4 pagine in totale: 1 2 3 4 zgt;'gt; Content article Home/Automation deployment infrastructure in the cloud with Terraform and Azure pipelines Terraform è uno strumento per creare, modificare e versione dell'infrastruttura in modo sicuro ed efficiente. Terraform è in grado di gestire i provider di servizi cloud esistenti e popolari e le soluzioni utente. I file di configurazione descrivono i componenti Terraform necessari per eseguire una singola applicazione o l'intero data center. Terraform genera un piano di esecuzione che descrive le modalità di esecuzione per ottenere lo stato desiderato e quindi lo esegue per creare l'infrastruttura descritta. Quando la configurazione cambia, Terraform può determinare cosa è stato modificato e creare piani di esecuzione aggiuntivi che possono essere applicati. Che cosa è coperto in questo laboratorio In questo laboratorio, si vedrà come strumenti open source come Può essere utilizzato per implementare l'infrastruttura come il codice (IaC) Come automatizzare la distribuzione dell'infrastruttura nel cloud con terraform e pipeline di Azure L'immagine successiva attraverso tutti i passaggi illustrati in questo laboratorio prima di iniziare a fare riferimento all'inizio della pagina prima di iniziare a seguire gli esercizi. Usare il generatore dimostrativo DevOps di Azure per fornire un progetto in DevOps di Azure. Questo URL selezionerà automaticamente il modello Terraform nel generatore demo. Se si vuole provare altri progetti, usare questo URL anziché il generatore azuredevops Seguire un semplice esempio passo-passo per imparare a usare il generatore dimostrativo di Azure DevOps. Esercizio 1: Esplorare il file Terraform (IaC) nel codice sorgente in questo laboratorio, si utilizzerà PartsUnlimited, che è un esempio di un sito Web di e-commerce sviluppato utilizzando .Net Core. Verrà esplorato un file terraform che ti aiuterà a fornire le risorse di Azure necessarie per distribuire il sito Web PartsUnlimited. Passare al progetto creato in precedenza usando il generatore dimostrativo Di Azure DevOps Select Repos. Passare al ramo terraforme. Assicurati di essere in un ramo terraforme e che la cartella Terraform si trova nel file. Selezionare webapp.tf file nella cartella Terraform. Passare attraverso il codice. webapp.tf è un file di configurazione terraform. Terraform utilizza il proprio formato di file denominato HCL (Hashicorp Configuration Language). È molto simile a YAML. In questo esempio, si vuole distribuire il gruppo di risorse di Azure, il piano di manutenzione dell'applicazione e il servizio dell'applicazione necessario per distribuire il sito Web. È stato aggiunto il file Terraform (Infrastruttura come codice) al repository del controllo del codice sorgente nel progetto DevOps di Azure, che può distribuire le risorse di Azure necessarie. Se vuoi saperne di più sulle basi di terraforma clicca qui. Esercizio 2: Creare un'app usando Azure CI Pipeline In questo esercizio si creerà l'app e si pubblicheranno i file necessari in un elemento denominato drop. Vai alle condutture - le condutture. Selezionate Terraform-CI e fate clic su Modifica (Edit). La vostra linea di assemblaggio sarà simile a quella qui sotto. Questa pipeline CI ha il compito di compilare il progetto .Net Core. Le attività nella pipeline ripristineranno le dipendenze, verranno compilate, verificate e pubblicate l'output dell'assembly nel file di posta (pacchetto) che può essere distribuito nell'applicazione Web. Per altri suggerimenti su come compilare progetti .Net Core con pipeline di Azure, vedere qui. Oltre a creare l'applicazione, è necessario pubblicare i file terraform per creare elementi in modo che siano disponibili nella pipeline del CD. Così abbiamo aggiunto l'attività Copia file per copiare il file Terraform nel catalogo Artefatti. Fare clic sulla coda per chiamare la compilazione. Una volta completata la compilazione, assicurarsi che gli elementi abbiano una cartella Terraform e PartsUnlimitedwebsite.zip file in drop. Esercizio 3: Distribuire risorse utilizzando Terraform (IaC) in Azure CD Pipeline In questo esercizio si creeranno risorse di Azure usando Terraform come parte della pipeline di distribuzione (CD) e si distribuirà PartsUnlimited al servizio dell'applicazione fornito da Terraform. Vai alle pipeline - qgt; rilasci. Selezionare Terraform-CD e fare clic su Modifica. Selezionare la fase Di sviluppo e fare clic su Visualizzazione attività fase per visualizzare le attività della pipeline. Vedrai attività come quella qui sotto. Scegliere Seleziona Il compito di CLI. Selezionare una sottoscrizione di Azure dall'elenco a discesa e fare clic su Autorizzazione per configurare una connessione di Azure. Per impostazione predefinita, gli archivi Terraform vengono mescolati localmente in un file denominato terraform.tfstate. Quando si lavora con Terraform in un team, l'utilizzo di un file locale rende più difficile l'utilizzo di Terraform. Quando è remoto, Terraform registra i dati sullo stato in un archivio dati remoto. In questo caso, usare l'interfaccia della riga di comando di Azure per creare un account di archiviazione di Azure e un contenitore di archiviazione Terraform. Per altre informazioni sullo stato remoto di Terraform, fare clic qui per selezionare Azure PowerShell. Selezionare la connessione di Azure dall'elenco a discesa. Per configurare il back-end Terraform, è necessaria una chiave per l'account di archiviazione. In questo caso, usare l'attività di Azure PowerShell per ottenere la chiave per accedere all'account di archiviazione fornito nella fase precedente. Scegliere l'attività di sostituzione dei token. Se si osserva webapp.tf file nell'Esercizio 1, nel passaggio 3 verranno visualizzati diversi valori con suffisso e con prefisso con Kew. Ad esempio, __terraformstorageaccount__. Usando l'attività di sostituzione del token, questi valori sostituiranno con le variabili definite nella pipeline di rilascio. L'installazione di Terraform viene usata per installare una versione specifica di Terraform da Internet o dalla cache degli strumenti ed è pre-utilizzata dall'agente pipeline di Azure PATH (posizionato o privato). Quando terraform viene avviato nell'automazione, l'attenzione è di solito sul piano/ciclo dell'applicazione principale. Il flusso di lavoro principale di Terraform è illustrato di seguito: i. Avviare il catalogo di lavoro Terraform. Creare un piano di modifica delle risorse in base alla configurazione corrente. Applicare le modifiche descritte nel piano. Le attività Terraform seguenti nella pipeline di rilascio consentono di implementare questo flusso di lavoro. Scegliere Terraform init. Selezionare la connessione di Azure dall'elenco a discesa. E non dimenticate di inserire il nome del contenitore come terraformi. Per altri parametri di attività, vedere questa attività qui come un comando terraform init. Il team terraform init esamina tutti i file nella directory di lavoro corrente e scarica automaticamente uno dei provider necessari per loro. In questo esempio, verrà caricato Azure man mano che vengono distribuite le risorse di Azure. Per ulteriori informazioni sul terraforme del team init, fare clic qui per selezionare l'attività del piano Terraform. Selezionare la connessione di Azure dall'elenco a discesa. Il comando terraform plan viene utilizzato per creare un piano di esecuzione. Terraform determina le azioni necessarie per ottenere lo stato desiderato specificato nei file di configurazione. Si tratta di una corsa a secco e mostra quali azioni saranno fatte. Per ulteriori informazioni sul team del piano terraform, fare clic qui per selezionare <a0>Applica</a0>. Selezionare la connessione di Azure dall'elenco a discesa. Questa attività funzionerà terraform per applicare il team per distribuire le risorse. Per impostazione predefinita, fornirà anche che si desidera applicare queste modifiche. Durante l'automazione della distribuzione, viene aggiunto un argomento di approvazione automatica in modo da non richiedere conferma. Conferma. La sfida della distribuzione del servizio app di Azure. Selezionare la connessione di Azure dall'elenco a discesa. Questa attività distribuirà il pacchetto PartsUnlimited nel servizio dell'applicazione Azure fornito da Terraform nelle fasi precedenti. Al termine, salvare le modifiche e creare una versione. Dopo un rilascio riuscito, passare ad Azure. Cerca pulterraformweb nei servizi app. Selezionare <a0>web-xxxx</a0> ed esaminare l'app. Vuoi saperne di più su Terraform? In tal caso, fare clic qui per la documentazione di Terraform. In questo laboratorio si è appreso come automatizzare le distribuzioni ripetitive con Terraform in Azure con pipeline di Azure. Guida È possibile guardare il seguente video che passa attraverso tutti i passaggi illustrati in questo laboratorio