All Day DevOps 2019 Report

Novembre 2019

Par Alexandre SCHWARTZMANN, Expert Cloud & DevOps FINAXYS

24 heures, plus de 150 sessions réparties sur 5 tracks (CI/CD, Cultural Transformation, DevSecOps, Cloud Native et SRE) : c'est le menu de la conférence AllDayDevOps qui s'est tenue le 6 novembre 2019 !

Le principe de cet événement hors norme est de faire intervenir des experts du monde entier sur ces sujets lors de sessions en ligne de 30 minutes réparties sur 24 heures.

Multi Cloud "Day to Day" - DevOps Power Tools

Par Ronen Freeman (Darillium)

On commence par une question assez simple, mais que beaucoup se posent : **comment survivre dans un contexte DevOps et multi-cloud ?**
En effet, ce n'est pas toujours évident, notamment si l'on a longtemps été habitué à des contextes en silo, car le DevOps demande la mise en pratique de nouvelles méthodes de travail.

Tout d'abord, quelques prérequis simples :

- Se renseigner/former sur les grands principes et les concepts de base de manière à savoir de quoi on parle.
- Préférer les équipes motivantes, parce que l'entraide et la collaboration c'est plus agréable.
- Identifier ses mentors, car ils permettent de définir des idéaux à atteindre et à se dépasser.

Ensuite, quelques qualités à cultiver :

- Une curiosité de tous les instants : se tenir au courant, se former, expérimenter
- Éviter au maximum de s'attacher aux technologies, car le monde de l'IT actuel change très rapidement
- Anticiper : c'est le meilleur moyen pour accompagner les changements
- « *Learn Fast, Adapt Fast* »

Une fois dans le bon état d'esprit, on peut commencer à parler d'outillage.
On peut répartir ces outils en 5 catégories :

Code	Package	Deliver	Platform		Monitor
			Infra	Config	
Bash	Docker	CI/CD	Cloud provider	Ansible	Splunk
Python	Maven	Helm	Kubernetes	…	Dartadog
…	…	…	Terraform		Prometheus
			…		

Il reste une dernière question : **dois-je automatiser ?**
Encore une fois, la réponse est assez simple.
Dans la philosophie DevOps, un paramètre important (le plus important ?) est **le temps**.
À partir de là, on peut transformer la question d'origine en :

« Le temps passé à automatiser est-il supérieur au temps gagné ? »

Il n'est donc plus tant question de « survivre » dans un contexte DevOps, que de s'y épanouir.

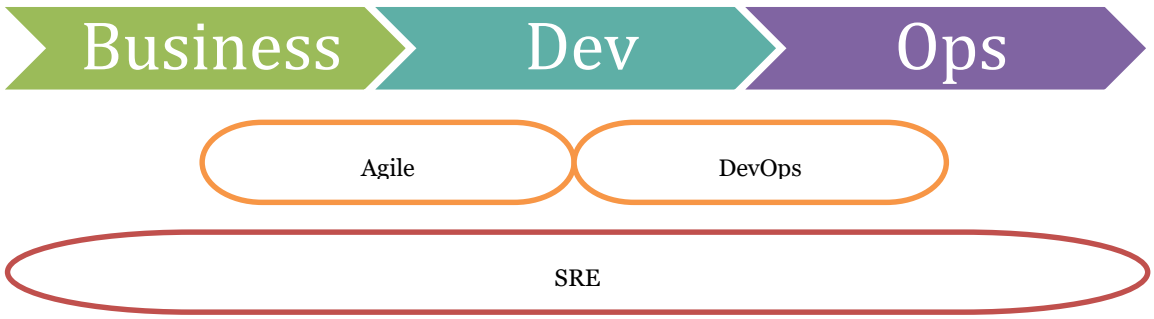
How To Run Smarter In Production - Getting Started With Site Reliability Engineering

Par Jennifer Petoff (Google)

Vous entendez peut-être déjà parler de SRE autour de vous, mais qu'est-ce donc ?
Il est vrai que c'est un concept qui commence à faire parler de lui, souvent en lien avec la philosophie DevOps.

Si on devait simplifier, la philosophie DevOps est l'extension des principes Agile et Lean des équipes de développement jusqu'aux équipes de production (ops).
Elle définit des grands principes, mais pas comment les implémenter.

SRE est une approche plus « globale » de l'IT, qui décrit une implémentation de ces principes depuis le Business jusqu'à la Production génératrice de valeur.



L'idée derrière tout cela est d'avoir une équipe de Site Reliability Engineers (ingénieurs SRE) qui vont agir selon plusieurs axes :

- Définir et suivre des SLO (Service Level Objectives)
  Ce sont des objectifs opérationnels basés sur des métriques (à ne pas confondre avec les SLA qui sont des objectifs contractuels)
  SLO atteints = des utilisateurs contents !

- Essayer d'être le plus fiable possible
  Attention cependant : 100% n'est pas un bon objectif de fiabilité à cause de la Loi de Murphy. Il faut rester réaliste.
  Prévoir un « Error Budget » pour prendre en compte les incidents et les problèmes qui surviennent. Cela comprend la charge nécessaire pour les actions de correction, les postmortems, etc…
  Limiter tant que possible le temps passé sur les actions sans valeur et favoriser leur automatisation

- Réguler la charge de travail en mettant en place un « Shared Responsability Model »
  L'équipe SRE ne remplace pas le service de Production.
  Une équipe surchargée est le signe que des améliorations sont possibles.

Automatiser tout ce qu'il est possible et pertinent d'automatiser.
Cela permet de :

- Éliminer les actions manuelles sources d'erreurs
- Mieux gérer son capacity planning
- Implémenter des mécanismes de résilience automatique

En conclusion, une équipe SRE jongle entre :

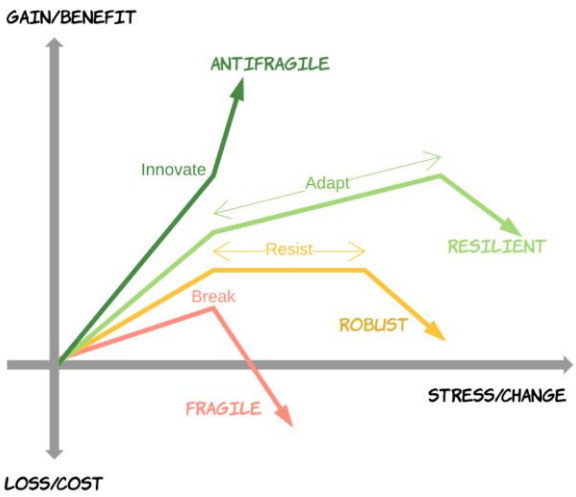
- La fiabilité de la Production et des pipelines de livraison
- La balance entre les temps de de développement, d'amélioration, de correction, etc…
- La vélocité
- Les coûts
-

The Dream Of The Antifragile Systems

Par Victor Martinez (Versia)

"I have a dream …"
Martin Luther King Jr (1963)

Dans le monde de l'IT, ce rêve est peut-être d'avoir des systèmes robustes, résilients, à toute épreuve.
Qui n'a pas un jour rêvé de s'affranchir de tous les soucis pour ne se concentrer que sur les activités apportant de la valeur ?
C'est de cette réflexion qu'est né le concept de système « Anti-fragile », définit dans le livre du même nom écrit par le professeur Nassim Nicholas Taleb.
L'anti-fragilité est définie comme fondamentalement différente des concepts de résilience (la capacité à se remettre d'un échec) et de robustesse (la capacité de résister à l'échec).
Au contraire, il est question d'anticiper pour réagir « positivement » à un stress, permettant ainsi d'éviter les effets « d'usure » :



Il existe plusieurs outils permettant de tendre vers ces systèmes, dont :

- L'AIOps
  L'utilisation de l'intelligence artificielle et du Machine Learning permet désormais, de manière accessible, d'étudier le comportement des systèmes et d'anticiper leur réaction à n'importe quel événement.
  Cela permet ensuite d'automatiser des actions correctrices (auto résilience), ou d'amélioration (antifragilité).

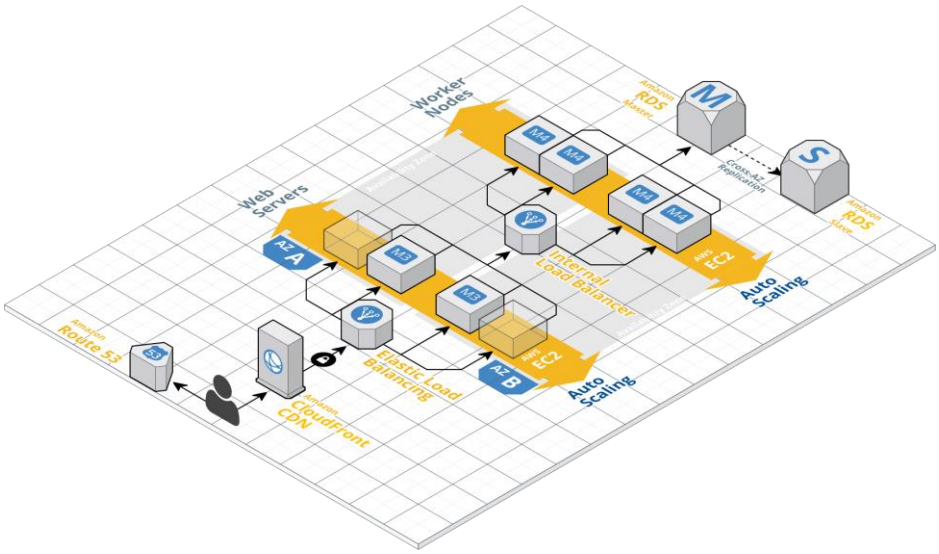
- Le Chaos Engineering
  Concept « inventé » par Netflix avec sa Simian Army (https://medium.com/netflix-techblog/the-netflix-simian-army-16e57fbab116), le Chaos Engineeing peut se résumer en « l'art d'introduire des comportements aléatoires pour valider la robustesse et la résilience de son système ».
  L'idée étant pour Netflix de détruire de temps en temps et aléatoirement des ressources (machine, zone, région, etc.) ou d'introduire des perturbations (latence par exemple) sur leur Production afin de vérifier en conditions réelles que le service est toujours rendu correctement.
  Le principal intérêt de cette méthode, au-delà de valider le bon fonctionnement de la Production en toutes circonstances, est de « forcer » les équipes à concevoir des nouvelles briques ou des nouveaux systèmes anti-fragiles by-design.

What You See Is What You Get For AWS Infrastructure

Par Anton Babenko (Betajob AS)

L'Infrastructure As Code est le fait de définir, de gérer et de déployer toute son infrastructure comme n'importe quel code applicatif. C'est-à-dire sous la forme de scripts versionnés, livrés et déployés exactement comme le ferait une équipe de développement.
On gagne une plus grande maitrise de son infrastructure, plus facilement auditable, mais on capitalise aussi avec l'utilisation des mêmes outils d'usine logicielle que ceux déjà utilisés par les développeurs.
Terraform est l'un de ces outils permettant de définir son infrastructure sous forme de scripts.

Dans cette présentation sous forme de démo, Anton Babenko va encore plus loin et nous montre comment on peut partir d'un schéma d'infrastructure pour créer ses plateformes sur AWS, le tout « as code ».
Tout commence avec le design de l'infrastructure souhaitée sur https://cloudcraft.co/ :



Une fois le schéma terminé, il suffit de l'exporter en code Terraform via le menu pour se retrouver avec un fichier zip contenant tous les fichiers utiles pour lancer le déploiement avec Terraform.
Ces fichiers sont générés en utilisant les librairies et les outils de https://modules.tf pour générer le code Terraform et Terragrunt automatiquement.
Il ne reste plus qu'à effectuer les dernières customisations éventuelles (dépendances, droits, scripts d'initialisation, etc.) et enfin, lancer le déploiement.
….
Sommes-nous en train d'assister aux prémices d'un nouveau concept, l'Infrastructure as Schema ?

Why Manual Verification Still Matters

Par Jeroen Willemsen *(Xebia)*

Jeroen Willemsen se questionne :

- Jusqu'où aller dans l'automatisation des aspects de sécurité ?
- Est-il possible de se passer de vérifications manuelles dans un pipeline DevSecOps ?

En effet, la philosophie DevSecOps pousse vers l'automatisation des différentes étapes du pipeline de livraison. Celui-ci incluant aussi les aspects « sécurité ». Les principes étant les suivants :

1. Toujours commencer par effectuer les validations manuellement avant d'automatiser. Car maîtriser les actions de vérification, ce sont les automatiser correctement et efficacement.

2. Toujours auditer la sécurité de son code, et faire de même avec toutes les dépendances. Vérifier le code c'est bien, revérifier constamment c'est mieux. Il suffit d'une montée de version non maitrisée d'une dépendance pour que se glisse en elle une faille de sécurité.

De la même manière, il faut s'assurer que l'on ne fait pas fuiter des informations (secrets dans le code ou dans l'image docker, etc …)

3. Au-delà de son propre code ou de ses dépendances, il ne faut pas oublier d'auditer la sécurité de la totalité des couches permettant à l'application de fonctionner :

Cela comprend les éléments suivants, sans s'y limiter : le matériel, l'OS, les librairies et applications, le runtime (python, java, docker, etc…), les conteneurs, la configuration, les couches d'exposition réseau, etc.
Chacun de ces éléments pouvant contenir des failles de sécurité.

4. Privilégier le principe du « moindre accès » pour tous les droits, que ce soit au niveau physique, qu'au niveau des droits RBAC.

5. Chercher la simplicité (principe KISS : *Keep It Simple, Stupid*), plus simple à maintenir et à auditer. Favoriser les conceptions « immutables » pour éviter les dérives entre les environnements.

Une fois tout cela bien maitrisé et automatisé, est-on certain d'avoir couvert tous les risques ?
Non.
Pour une raison assez simple : ce n'est pas parce qu'il n'y a pas de CVE qu'il n'y a pas de faille : il suffit qu'elle n'ait pas encore été détectée, identifiée ou simplement remontée.

C'est là où il est nécessaire d'avoir une validation manuelle. Que ce soit dans la rédaction de tests customs dont les outils du marché sont probablement dépourvus, voir de cas trop absurdes pour avoir été prévus.

Building Modular Infrastructure in Code

Par Fergal Dearle

Construire ses infrastructures « as code » c'est garantir une meilleure maitrise de ce qui est provisionné, mais encore faut-il éviter de tomber dans les mêmes travers que les équipes de développement.
Comment se prémunir du code en double, des équipes qui réinventent la roue, etc … avec toutes les conséquences que cela implique (duplication, mauvaise maintenabilité etc. ?

La solution, c'est de rendre son code le plus modulaire et réutilisable possible.

Pour ce faire, la bonne méthodologie est de :

1. Identifier ses différentes couches.
2. Identifier les blocs et les archétypes.
3. En déduire des patterns.
4. Coder ces patterns en « fonctions » réutilisables.

Ces « fonctions » utilisant le système d'export pour se passer des valeurs entre elles.
De la même manière, il est conseillé d'externaliser de ces fonctions tout script et toute configuration (dans un repository de config par exemple)
Enfin, ne pas oublier de tester !

Il faut néanmoins faire attention et anticiper les potentiels problèmes suivants :

- La dérive dans le temps des stacks déployées ainsi (le drift).
- Les problèmes de compatibilité durant tout le cycle de vie.
- Les problèmes d'incompatibilité entre les différentes régions ou differents cloud providers.

There's no nice way to say this: Your DevOps has gone horribly wrong

Par Kalle Sirkesalo (Eficode ROOT)

La philosophie DevOps est formidable et permet grandement d'améliorer la qualité et les délais de mise en production de nouvelles fonctionnalités générant de la valeur.
Oui, mais nous avons tous vécu ou entendu parler de cas où la situation n'était pas si rose.
En effet, il faut éviter certains écueils récurrents :

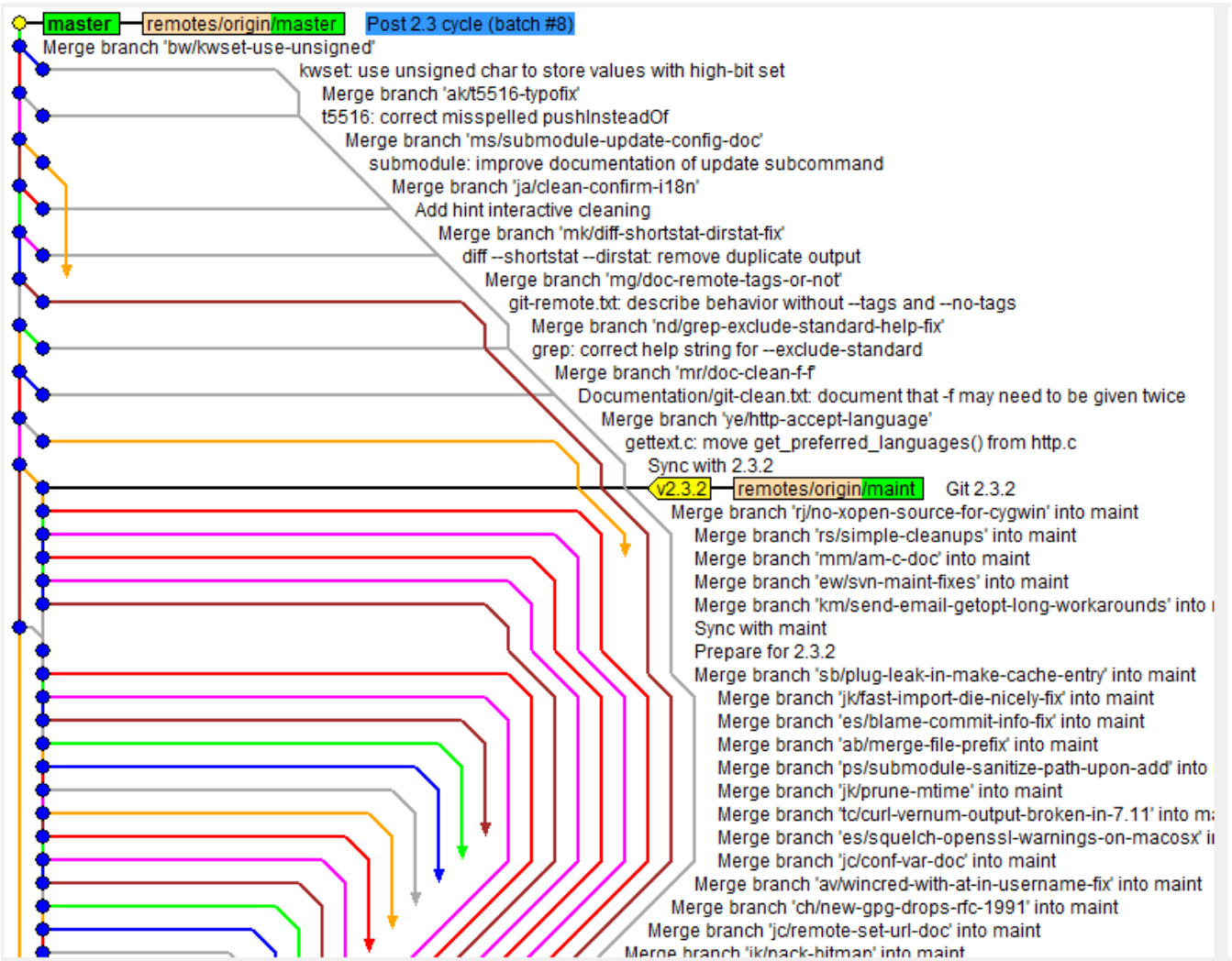
- Se focaliser à outrance sur les outils.
- Croire que le pipeline CI/CD remplace les rétros et les démos.
- Penser que la vélocité est proportionnelle au nombre de personnes dans les équipes.
- Imaginer que l'on a plus besoin de la priorisation du business.

Qu'est ce qui pourrait mal se passer ?
Un cas qui arrive souvent est d'avoir :

- Trop d'équipes de développement travaillant sur des sujets liés.
- Trop d'environnements de développement, complexe à gérer.
- Trop de branches de développement (durant souvent depuis trop longtemps), difficile à synchroniser.

On se retrouve alors avec un process de release management infernal, des environnements nombreux chacun avec une version différente, etc …

Ce qui conduit fatalement à un pipeline complexe, long et humainement impossible à maitriser.



Est-ce que ça pourrait être pire ?
Oui.
Surtout si en plus de toutes ces mauvaises pratiques techniques on ajoute une couche organisationnelle qui n'est pas adaptée au DevOps.

Quelques exemples :

- Des équipes dépendantes n'utilisant pas la même méthode ou les mêmes principes (Agile/kanban/waterfall/…), rendant non optimal voire impossible la synchronisation des actions.
- Un enfer organisationnel, si celle-ci n'est pas en accord avec les responsabilités de chaque rôle. Par exemple, si le PO ne dépend pas de la même hiérarchie que les équipes.
- Une priorisation infernale, s'il n'y a pas d'objectif commun.

Peut-on s'en prémunir ?
Heureusement oui, en suivant quelques principes de bon sens :

- Le PO et les équipes de développement et les ops doivent être dans la même « entité ».
- Utiliser des branches de développement courtes : « code fast »
- Garder un minimum d'environnements de développement, pour un pipeline court et efficace
- Prioriser le backlog des tâches, plutôt que de faire plusieurs choses en même temps
- Communiquer, communiquer et communiquer
- Ajouter des équipes ne permet pas de gagner en vélocité (« *on ne peut pas faire un enfant en 4 mois et demi avec 2 femmes* »)

Mon retour d'expérience :

Le + : des intervenants de qualité et le découpage en 5 tracks variés :

- CI/CD
- Cultural Transformation
- DevSecOps
- Cloud Native
- SRE

Comme à chaque fois, je ressens un sentiment de frustration de n'avoir pas pu suivre tous les talks car il a fallu choisir entre les 5 tracks de chaque session.
Particularité encore plus frustrante du **AllDayDevOps** du fait de son format 24h, c'est de devoir aussi faire l'impasse sur les live talks se déroulant à des horaires peu raisonnables.

N'oubliez pas : restez curieux ! Et rendez-vous l'année prochaine pour une nouvelle édition.

--- Par Alexandre SCHWARTZMANN, Expert Cloud & DevOps FINAXYS

Retrouvez toutes les sessions en replay sur ces liens :

ADDO Channel 1 : Block 1 https://www.youtube.com/watch?v=sHNgvs_lVzg
@1:52: devsecops with owasp devslop
@31:42 anatomy of a continuous delivery pipeline
@1:01:28 devsecops for microservices with resilience
@1:30:39 everybody gets a staging environment
@1:59:00 CICD for fintech

@3:00:10 CICD for serverless application on AWS

ADDO Channel 1 : Block 2 : https://www.youtube.com/watch?v=Nu5KLZwmYRI
Kalle Sirkesalo: There's No Nice Way to Say This: Your DevOps Has Gone Horribly Wrong.
Dmitri Lerko: A Year With GitOps In Production: Retrospective.
Angel Rayo: Azure DevOps From Start To Star.

ADDO Channel 1 : Block 3 https://www.youtube.com/watch?v=zc9z2_ZwP4o
Adarsh Shah: Integrating Infrastructure As Code Into A Continuous Delivery Pipeline.
David Maillet: Delivery Flexibility - Pipeline As Code With Jenkins & Groovy.
Sasha Rosenbaum: CI/CD For Machine Learning

ADDO Channel 1 : Block 4 : https://www.youtube.com/watch?v=EXMND7WCiLc
Tim Davis: Being Budget Conscious In A Continuously Automated World.
Chaim Sanders & Franziska Buehler: The OWASP Core Rule Set - Crafting A Security Focused DevOps Program
Mae Large & Russ Parmer: GitOps FTW
Nora Kennedy: Continuous Testing – Preparing Integration Data
Maarika Krumhansl: Embracing On-Prem Delivery With DevOps

ADDO Channel 1 : Block 5 : https://www.youtube.com/watch?v=56aA3A9igvw
Madison Cool & Chris Smith: An Intelligent Approach To Upgrading OSS Libraries
Rosalind Radcliffe: Open Pipeline & Mainframe, Yes They Go Together

ADDO Channel 2 : Block 1 : https://www.youtube.com/watch?v=-JuPprlGb48
@1:02:39 Devops Assurance With OWASP SAMMv2
@1:27:34 The (S)SDLC Making the web secure by design++
@2:23:23 The OWASP ZAP HUD
@2:51:18 Why manual verification still matters
@3:22:13 Compliance as code
@3:57:07 OWASP Juice Shop

ADDO Channel 2 : Block 2 : https://www.youtube.com/watch?v=D38NGcjLH_g
Keynote Session: Simon Wardley, Crossing.
The River By Feeling The Stones.
Judy Johnson: The Science of Compliance: Early Code To Secure Your Node.
William Zhang: The World Bank Group's Cloud
Journey With DevSecOps.
Tony UcedaVelez: Getting
Your Security Program to Shift Left - Operationalizing Security Controls via DevSecOps

ADDO Channel 2 : Block 3 : https://www.youtube.com/watch?v=aIHaLiUw8oM
Keynote: Patrick Debois, DevOpsDays: Hints & Glimmers Of Things To Come.
Tom Stiehm: Shifting Security Left: The Innovation of DevSecOps.
Sean Davis: Breaking Bad - DevOpsSec To DevSecOps
Rich Mills: DevSecOps - Pipeline Tooling For Continuous Security In Agile Projects
Kumar Mathialagan & Shiva Nouzari: Supply Chain Security - Where, What, How

ADDO Channel 2 : Block 4 : https://www.youtube.com/watch?v=zTRN120Uu28

Keynote: Kevin O'Dell and Jeremy Castle, State Farm - Damming The $100 Billion Waterfall: Transforming The Enterprise To DevOps At State Farm.
Manuel Pais: Beating The 1:100 Odds - Team Design For Security.
John Melton: OWASP AppSensor - Self-Defending Applications Through Real-Time Event Detection & Response.
Jeffrey Payne: Lessons Learned Implementing DevSecOps.
DJ Schleen: Imperial vs. Metric & Why Deft Density Sucks Aaron Rinehart: Security Pre-Cognition: DevSecOps & Chaos Engineering

ADDO Channel 2 : Block 5 : https://www.youtube.com/watch?v=gOLQ-7tL4v8
Keynote: Emily Freeman, Microsoft: The Intersection of Communication & Technology
Alexi Karuna: Brownfield Terraform Implementation On Multiple AWS Regions (Twitter's SnappyTV)
Anjlica Malla: Cloud Migration Security
Boyd Hemphill: A History of Dev*Ops
Jennifer Czaplewski: Security Data - GPS For Application Teams
Matt Tesauro: Running FaaS With Scissors

ADDO Channel 2 : Block 6 : https://www.youtube.com/watch?v=qu9_HyW2uRM
Edwin Kwan: Keeping Up With Security - An Automated, Self Service Approach
Hossam Barakat: Secure Your Kubernetes Containers
Abhisek Datta: Application Security Workflow Automation Using Docker & Kubernetes
Akash Mahajan: Reliable & Automated Cloud Native Security Operations
Madhu Akula: Breaking & Pwning Docker Containers & Kubernetes Clusters
Vandana Verma: Taking Away The Sugar & Giving The Pile To Ants

ADDO Channel 3 : Block 1 https://www.youtube.com/watch?v=lZTMkEA7jws
Kris Buytaert: The History Of DevOps.
Laurie Barth: A Software Engineer's Guide To DevOps.
Rene van Osnabrugge: Growing Your DevOps Mindset.
Thiago de Faria: AI With A Devops Mindset: Experimentation, Sharing & Easy Deployment Of ML Components.
Vlatko Ivanovski: DevOps Metrics - Measuring What Matters.
Jason Gwartz: A Developer Team that Plants Trees.

ADDO Channel 3 : Block 2 : https://www.youtube.com/watch?v=jn8Xv8SiisM
Charlene Li: The Disruption Mindset - Why Some Organizations Transform While Others Fail.
Jayne Groll: Upskilling DevOps.
Eliza May Austin: WTF Is DevSecOps?

ADDO Channel 3 : Block 3 https://www.youtube.com/watch?v=Zyj0DgnseEg
Jamal Walsh: Influencing A DevOps Culture Across Distributed Teams & Organizations.
Marc Cluet: Managing DevOps.
Teams: Staying Alive Wendy Ng: What Is DevSecOps?
Enrique Carbonell & Jorge Pais: Automate Or NOT From The Beginning, That Is The Question...

ADDO Channel 3 : Block 4 : https://www.youtube.com/watch?v=khFkvYIFiIA
Aimee Bechtle: Pragmatic DevOps.
Carmen DeArdo: How Value Stream Management Will Transform IT & Business.

Filipp Kofman: Recent Developments In Open Source Licensing.
Shaaron Alvares: Project To Product - Managing Dependencies Across DevOps Teams With Story Mapping
Hasan Yasar & Nicolas Chaillan: DevSecOps Journey In DoD Enterprise

ADDO Channel 3 : Block 5 : https://www.youtube.com/watch?v=uymSPlasd6U
Josh Atwell: How To Be a Failure & Still Succeed
Ola Chowning: Skill Shift - From Deep To Wide, Hard To Soft
Troy DuMoulin: The Marriage Of DevOps Continuous Delivery & ITIL Transition - Decoding The Gordian Knot
Mykel Alvis: Blue Collar Knowledge Work
Leon Garcia & Gibran Lemus: Software Development With A Lean Mindset & DevOps

ADDO Channel 3 : Block 6 : https://www.youtube.com/watch?v=nYTGKeVzUBs
Fabian Iannerella: DevOps Is A Journey - Choose Your Own Adventure
Martin Alfke: DevOps In A Containerized Word
Magda Chelly: How To Start The Change For A DevOps Team In A Large Enterprise
Mirco Hering: New Barriers For DevOps Adoption
Niladri Choudhuri: ITIL4 & DevOps - A Way Forward For Dev & Ops
Liat Palace: Apply Genetic Engineering To Your Organizational Culture
Anton Weiss: Many Changes, Little Fun - Time To Measure The Value Of DevOps

ADDO Channel 4 : Block 1 https://www.youtube.com/watch?v=xtNx1t6lxQE
@9:07 Can Kubernetes Keep A Secret?
@31:16 Bring back the power of the ide
@1:01:52 What you see is what you get for aws infrastructure

ADDO Channel 4 : Block 2 : https://www.youtube.com/watch?v=Xzbk2pt0ilk
Ariane Gadd: Deploying Microservices To AWS Fargate.
Sean O'Dell: Automate Everyday Tasks With Functions.
Jesse Butler: Imposter Syndrome Ain't Just A River In Egypt.

ADDO Channel 4 : Block 3 https://www.youtube.com/watch?v=PVli8qvXgkM
Reza Rahman: Effective Docker & Kubernetes For Java EE Developers.
Bill Shetti: Creating A Stateful Application With K8S & AWS DB Services (Document DB/ElasticCache).
Mickey Boxell: Open Source Observability Tools.
Shrivatsa Upadhye: The 3 Muskeeters - Jenkins, Terraform, Vault

ADDO Channel 4 : Block 4 : https://www.youtube.com/watch?v=YS0Axx5SggY
Dan Barker: Making "Just Culture" Just Your Culture.
Chris Roberts: Holding The Industry Accountable…
Lee Calcote: Establishing An Open Source Program Office
Dawn Parzych: Working From the Beach & Other Myths Of Remote Employees
Dominica DeGrandis: Making Better Business Decisions With Flow Metrics

ADDO Channel 4 : Block 5 : https://www.youtube.com/watch?v=Bs-rzOmxa48
John Rauser: Beyond the Bottleneck - A New Theory Of Constraints
Martin Brunke: Transformation At The S.P.E.E.D. Of DevOps

Matthew Gabavics: Practical Tips For Making DevOps A Culture, NOT A Role
Hossam Barakat: Kubernetes For Developers

ADDO Channel 5 : Block 1 : https://www.youtube.com/watch?v=ike3vQbE4zc
@0:23 Multi Cloud "Day-to-Day" DevOps Power Tools
@30:17 How to Run Smarter in Production
@1:01:12 The Dream of the Anti-fragile Systems
@1:26:19 Autoscaling Services On All Dimensions
@1:58:03 Challenges In Implementing SRE In A Large Organisation

ADDO Channel 5 : Block 2 : https://www.youtube.com/watch?v=WXINx5rEl4Q
Molly Struve: Creating A Scalable Monitoring System That Everyone Will Love.
Ryan Lockard: The Enterprise SRE (eSRE) Approach.
Tom Leaman: Practice Makes Perfect - Developing Expertise Through Chaos Engineering.

ADDO Channel 5 : Block 3 : https://www.youtube.com/watch?v=a5J3CRWSOXA
Robbie Daitzman: Things Break, But How Do We Move Forward?
David Blank Edelman: How To Fail With SRE
Christina Yakomin: Observability Made Easy
Carlos Nunez: The Relationship Between SRE & BDD

ADDO Channel 5 : Block 4 : https://www.youtube.com/watch?v=N_A2uJzVZ8c
Chad Todd: Learning From Failure & How You Can Too.
Dan Illson: Fences & Gates - Designing Ops For DevOps
Jason Yee: Chaos - Breaking Your Systems To Make Them Unbreakable
Ram Lakshmanan: Troubleshooting Performance & Availability Problems
Kelley Arthur: The Hackers & The Attackers

ADDO Channel 5 : Block 5 : https://www.youtube.com/watch?v=6GO9h1VsXc4
Wendy Nather: ZeroTrustOps - Securing At Scale
Rebecca Wynn: Global Privacy & Security by Design (GPSbyDesign) & Default
Caroline Wong: OWASP Top 10 Overview
Michael Fraser: Modern Solution Delivery: IT As Code
Lukasz Radosz: DevSecOps Enabled Micro-Perimeter API Protection

ADDO Channel 5 : Block 6 : https://www.youtube.com/watch?v=HmjvBIuC-BQ
Christian Melendez: Development Workflow in Kubernetes
Suzanne Dyke: Cyber Education, Awareness & Influence
Tanvi Bali & Paul Arana: Practical DevSecOps - Tales from the Trenches Kaslin Fields: A Geek's Guide To DevOps Cultural Transformations