

A photograph of a person's arm and hand writing on a notepad at a wooden table. Another person's hands are visible in the background, gesturing. The image has a blue tint.

MINDSET

Experience the Human-Centered Enterprise

September 2022

Responding to Change: Agile SAP and DevOps

Presented by Andy Prier

MINDSET





Our Mission

Mindset is committed to making the technology experience significantly better at work for everyone.



Today's Agenda

- DevOps and Other Ops
- Getting to DevOps
- Industry Leaders for Change Practices
- Next Steps



DevOps and Other Ops

A shared definition

| *Ops

- DevOps
- DevSecOps
- DesignOps
- NoOps
- GitOps
- CloudOps

DevOps

DevOps is a set of cultural concepts focused on ways of building our technology systems.

It focuses on shared approaches to improving

- Collaboration
- Communication
- Automation
- Tooling
- Resiliency
- Transparency

Cultures are inherently human. With DevOps, we focus on building the culture through

- Technology
- Process
- Practice



Getting to DevOps

The journey is not turnkey...

Maturity Levels

Category	First Steps	Beginner	Intermediate	Advanced	Expert
Communication	Development, Security, Infrastructure, QA, Design on seperate teams communicating via meetings.	More informal cross-team communication and knowledge-sharing on demand.	Developing shared architecture and governance processes. May convene task forces from across teams.	Teams work collaboratively without immediate prompting. Strong shared architecture and governance within the group.	Full app life-cycle under one team with shared architecture and governance processes across company.
Agile Process	Most actions are ad hoc or based on the approach an individual uses.	Beginning to institute shared process and automation. Adopting some form of agile or iterative process.	Some processes are documented. Processes often remain frictional. Some resentment of process exists.	Most processes are documented and many are automated. Processes are well-shared between teams.	All primary processes documented. Processes are low friction, highly automated, and fairly well followed, not resented.
Automation	Builds, promotion, testing, are all manual.	Beginning to automate some aspects of development and deployment process.	Some automated testing. Deploy to Production is automated. Automation infrastructure exists but may be project-specific.	Aspects of development process are fully automated. Testing is on the way to maximal automation. Shared automation infrastructure.	No manual steps in build, sign-off, or deploy through Production deploy. Regression testing is automated.
Infrastructure	All systems installed on bare metal, manual setup and maintenance	Beginning virtualization and infrastructure automation. Patch application, QA rebuild, happen mostly regularly.	Virtualization is used throughout landscape. Regular support pack & refresh schedule. Some infrastructure automation.	QA/Dev refresh happens constantly and automatically. Beginning to use containerization. Support pack application automated.	Dev through Production containerized and available to developers to run. Support packs applied at least monthly.
Systems Support	Little to none beyond what comes with systems. People spend a lot of time in SE03, STMS transactions.	Integration based on manually maintained data (e.g. ticket numbers maintained as transport attributes)	Multiple internal development platforms (Issue Management, Agile, Source Control) with some degree of integration.	Interdependency is starting to be automated. E.g. Tickets result in development branches and transports being created automatically. Partial reporting transparency	Internal development platforms are highly interlinked, easy to navigate between. High degree of transparency across platforms

| Specific goals we'll focus on today

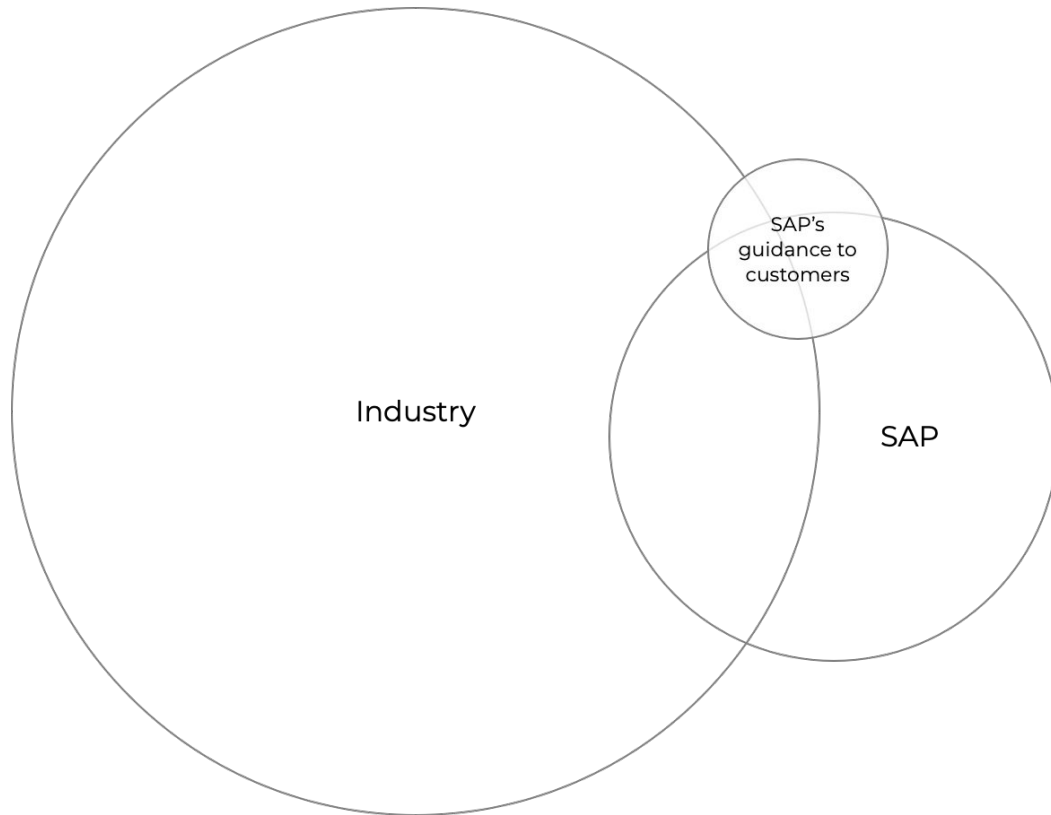
- Control your source
- Integration design into your development
- Reliable builds
- Secure builds
- Tested builds



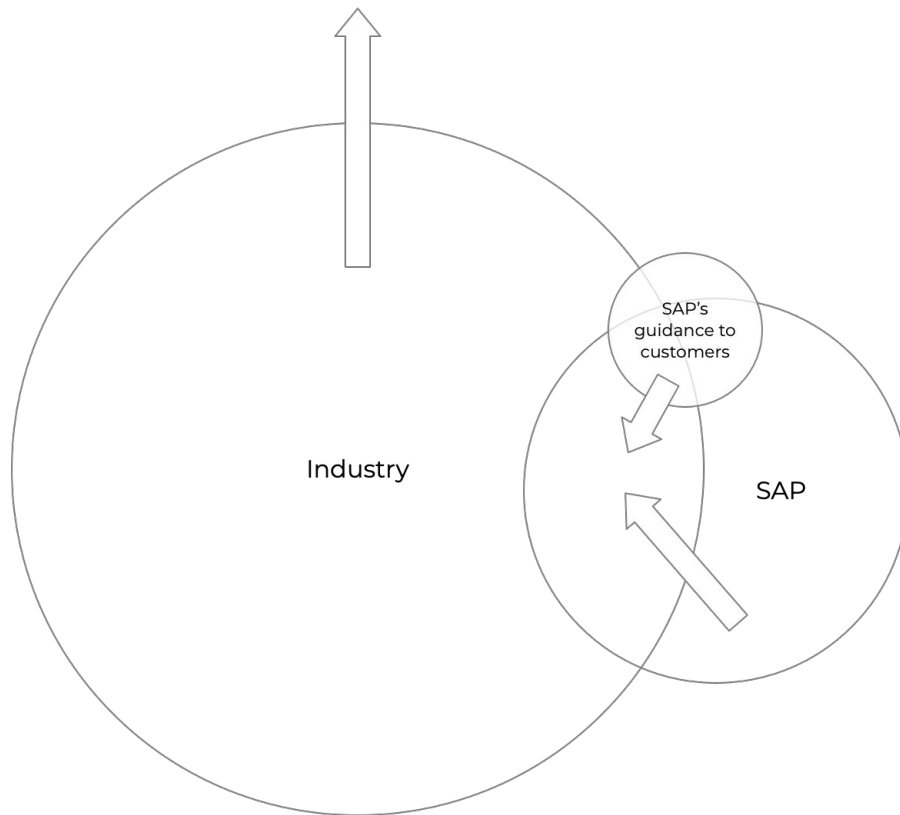
Industry Leaders for Change Practices

Looking away from our north star

DevOps Approach and Expertise



DevOps Velocity of Growth





Next Steps

Starting in the direction we want
to continue

| Next steps

- **Control your source:** Run your own Git
- **Design integration:** Designers embedded directly in teams
- **Reliable builds:** CI and automating all the things
- **Secure builds:** Control your package repository centrally
- **Tested builds:** Next steps in testing
 - Write functional tests
 - Static code analysis
 - Does it run the same? Identifying small % changes.
 - Accessibility testing

| Control your source - Really using Git

Joel on Software - [Distributed Version Control is here to stay, baby](#)

‘ In that podcast, I said, “To me, the fact that they make branching and merging easier just means that your coworkers are more likely to branch and merge, and you’re more likely to be confused.” ’

- Version Control - Like Subversion
 - Subversion tracks changes, and everytime your check it in, new copy
 - Therefore pulling a new project will take forever
 - It's still just a bunch of copies, BUT we can branch and merge. But what if 3-4 of us are using it?
- Git - Differences
 - Work disconnected with a local copy of the repo = **Resilient**
 - Git will track changes, but leave the unchanged lines, as is = **Smaller & Faster**
 - My working copy allows me to revert/merge without going to the server = **Distributed**
 - Every commit is cryptographically hashed = **More Secure**

| Design integration - Integrating design & dev tools

Our developers are focused on executing great design, but they have a lot of things on their minds. Can we help devs and designer to ensure we are delivering the design as intended?

Some tools we use a lot:

- Figma - our primary design tools
- Storybook - a UI component explorer
- Jira - our agile management system

Reliable builds - Controllable environment

Package.json scripts

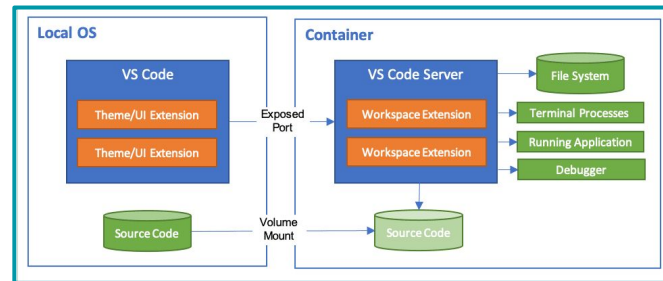
- More reliable builds
- Still suffers from dev dependency version issues between build environments

Build, develop, and test in containers

- Allows the development team to collaboratively control the entire dev/build/test environment
- Local tests are guaranteed to match CI/CD tests
- [Developing inside a Container using Visual Studio Code Remote ...](#)

```
"scripts": {  
  "start": "ui5 serve --config=ui5module/ui5.yaml --open index-bas.htm",  
  "build:ui": "run-s build:uimodule",  
  "test": "run-s lint karma",  
  "karma-ci": "karma start karma-ci.conf.js",  
  "clearCoverage": "shx rm -rf coverage",  
  "karma": "run-s clearCoverage karma-ci",  
  "lint": "eslint .",  
  
  "deploy": "npm run build:ui && fiori deploy --config ui5module/ui5.yaml",  
  "deploy-config": "fiori add deploy-config",  
  "serve:uimodule": "ui5 serve --config=ui5module/ui5.yaml",  
  "build:uimodule": "ui5 build --config=ui5module/ui5.yaml --clean-dist",  
},
```

📁 .devcontainer	Merg
📁 ui5module	Depl
📄 .editorconfig	Initia



Secure builds - Control dependency versions

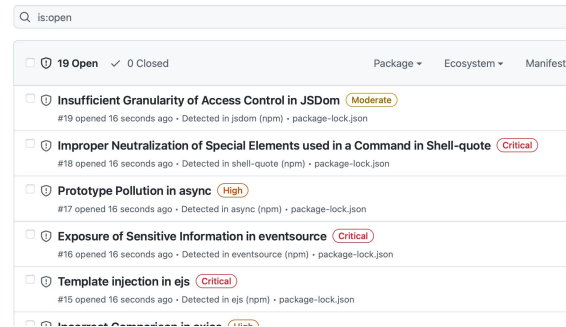
What is software supply chain security?

Simple steps:

- **"react": "^18.2.0" to "react": "18.2.0"**
- Dependabot or similar tools

Keep in mind: Front-end applications are not trusted applications, so security investments have limited return. The important thing to concentrate on is backend service and infrastructure security as well as developer education.

Dependabot alerts



is:open	
<input type="checkbox"/> 19 Open	<input checked="" type="checkbox"/> 0 Closed
Package	Ecosystem
<input type="checkbox"/> Insufficient Granularity of Access Control in JSDom	Moderate
#19 opened 16 seconds ago • Detected in jsdom (npm) • package-lock.json	
<input type="checkbox"/> Improper Neutralization of Special Elements used in a Command in Shell-quote	Critical
#18 opened 16 seconds ago • Detected in shell-quote (npm) • package-lock.json	
<input type="checkbox"/> Prototype Pollution in async	High
#17 opened 16 seconds ago • Detected in async (npm) • package-lock.json	
<input type="checkbox"/> Exposure of Sensitive Information in eventsource	Critical
#16 opened 16 seconds ago • Detected in eventsource (npm) • package-lock.json	
<input type="checkbox"/> Template injection in ejs	Critical
#15 opened 16 seconds ago • Detected in ejs (npm) • package-lock.json	
<input type="checkbox"/> Incorrect Command Injection	Critical
#14 opened 16 seconds ago • Detected in ... (npm) • package-lock.json	

Tested builds - Basic testing

Start small:

- Does it build
- Does it lint
- Does it run (bonus)

Can you run the equivalent of this Azure DevOps pipeline for every pull request/commit on every application you maintain?

```
1  # Node.js
2  # Build a general Node.js project with npm.
3  # Add steps that analyze code, save build artifacts
4  # https://docs.microsoft.com/azure/devops/pipelines
5
6  trigger:
7    - main
8
9  pool:
10     vmImage: ubuntu-latest
11
12  steps:
13    - task: NodeTool@0
14      inputs:
15        versionSpec: '10.x'
16        displayName: 'Install Node.js'
17
18    - script: |
19        npm install
20        displayName: 'npm install'
21
22    - script: |
23        npm run lint
24        displayName: 'ESlint'
25
26    - script: |
27        npm run build:ui
28        displayName: 'build'
```



Andy Prier

andyprier@mindsetconsulting.com



@andyprier

Appendix




Learning more

Resources and methods for *really*
learning how to do DevOps

Reference

- [What is DevOps](#) - Ernest Mueller, 2010
- [Distributed Version Control is here to stay, baby](#), Joel Spolsky, 2010
- [Developing inside a Container using Visual Studio Code Remote ...](#)
- [Compliance in a DevOps Culture](#) - Martin Fowler
- [Tagged supply chain](#) - Schneier on Security
- [Atlassian Git Tutorials](#)



member of the  

AppHaus network

Our Solutions and Products

