Layers of Leadership Tension

Lee Campbell

-25yrs	-20yrs	-15yrs	-10yrs	-5yrs
Product builder	Contractor	Consultant	Contributor	Product leader
.com	Web2.0	FinTech	DDD	*A.P.C ?*

LeeCampbell.com

Leadership Challenges	Start Up Challenges	Direction
Teamwork Challenges	Maps and Measures	Org Challenges
Driver Trees	Paths and Practice	Coordinated Strategy

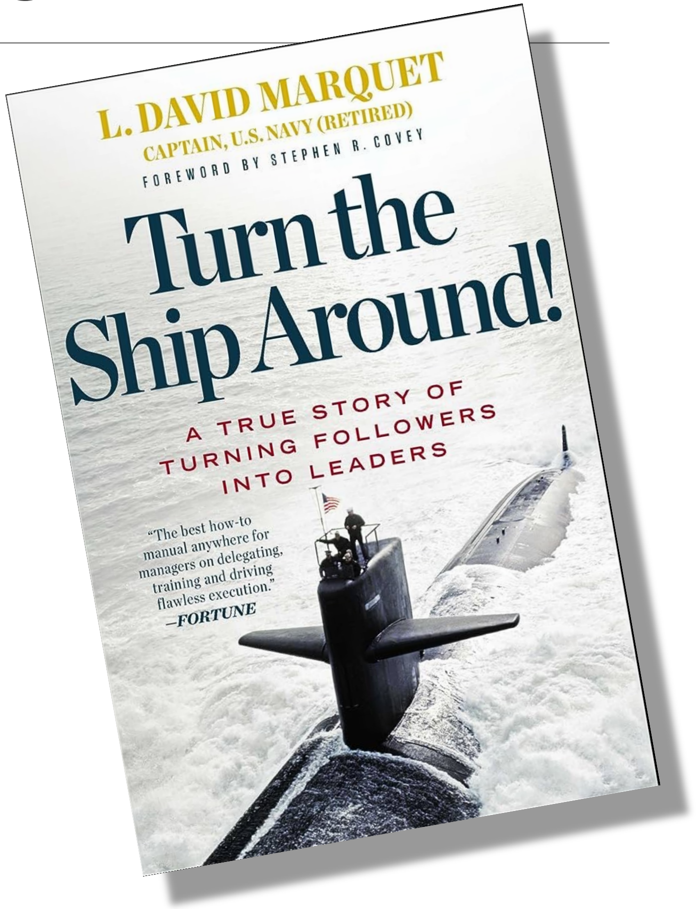
Part 1
Leadership Challenges

Product Leadership Challenges

- Working on different things
- Working different ways
- Variation in skills
- Compensating behaviours

Product Leadership Challenges

- Working on different things
- Working different ways
- Variation in skills
- Compensating behaviours
- **Control without competency is Chaos**


L. DAVID MARQUET
CAPTAIN, U.S. NAVY (RETIRED)
FOREWORD BY STEPHEN R. COVEY
Turn the Ship Around!
A TRUE STORY OF TURNING FOLLOWERS INTO LEADERS
"The best how-to manual anywhere for managers on delegating, training and driving flawless execution." —FORTUNE

Team Leader Concerns

- People
- Delivery
- Operations

Team Leader Concerns

- People
- Delivery
- Operations

- *Revenue capability*
- *Revenue potential*
- *Revenue actualization*

Team Success

ISSUES

- Being trusted
- Being valued
- Being aligned

Team Success

ISSUES

- Being trusted
- Being valued
- Being aligned

ACTIONS

- *Build it right, run it right*
- *Right thing, right time*
- *Right people, right place*

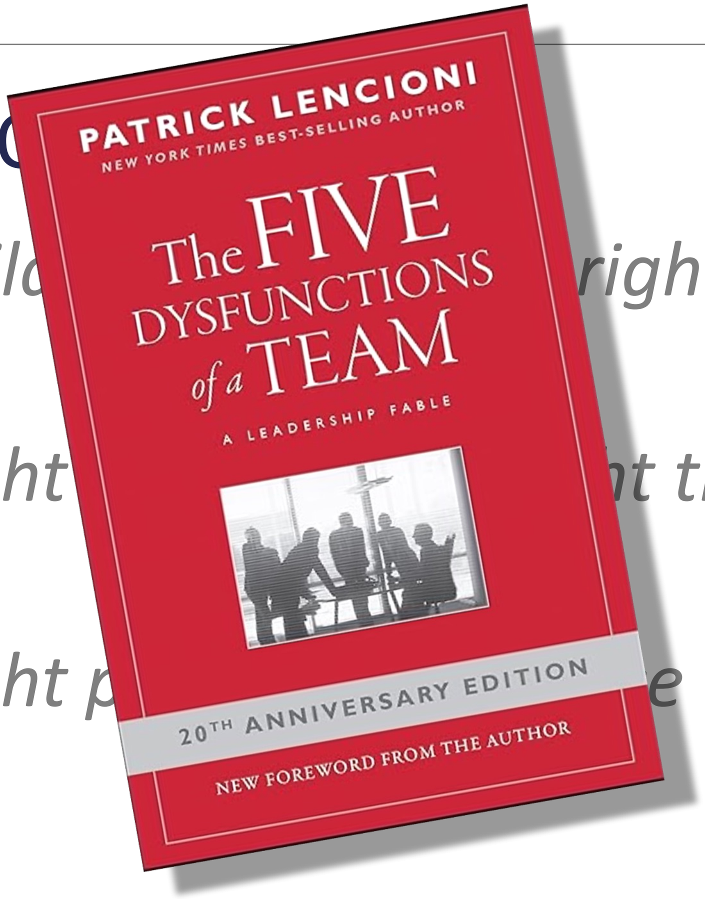
Team Success

ISSUES

- Being trusted
- Being valued
- Being aligned

ACTIC

- *Builc* *right*
- *Right* *t time*
- *Right p* *e*


PATRICK LENCIONI
NEW YORK TIMES BEST-SELLING AUTHOR
The FIVE DYSFUNCTIONS of a TEAM
A LEADERSHIP FABLE
20TH ANNIVERSARY EDITION
NEW FOREWORD FROM THE AUTHOR

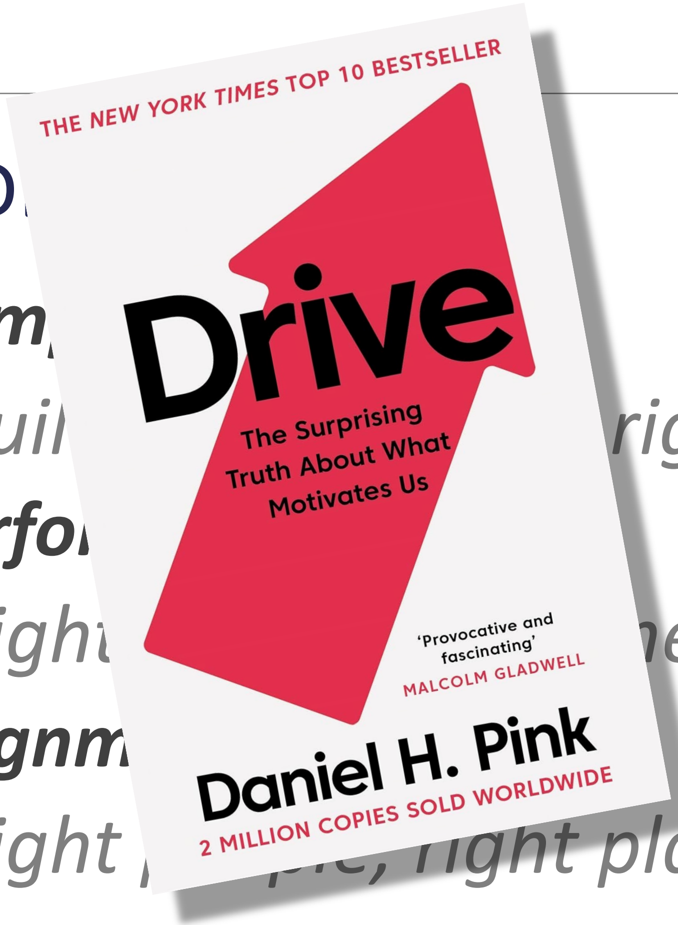
Team Success

PEOPLE'S ISSUES

- Being trusted
- Being valued
- Being aligned

LEAD

- ***Com***
  - *Buil right*
- ***Perfo***
  - *Right ne*
- ***Alignm***
  - *Right ple, right place*


THE NEW YORK TIMES TOP 10 BESTSELLER
Drive
The Surprising Truth About What Motivates Us
'Provocative and fascinating'
MALCOLM GLADWELL
Daniel H. Pink
2 MILLION COPIES SOLD WORLDWIDE

Team Success

PEOPLE’S ISSUES

- Being trusted
- Being valued
- Being aligned

LEADERS’ ISSUES

- ***Competency***
  - *Build it right, run it right*
- ***Performance***
  - *Right thing, right time*
- ***Alignment***
  - *Right people, right place*

Competency

Performance

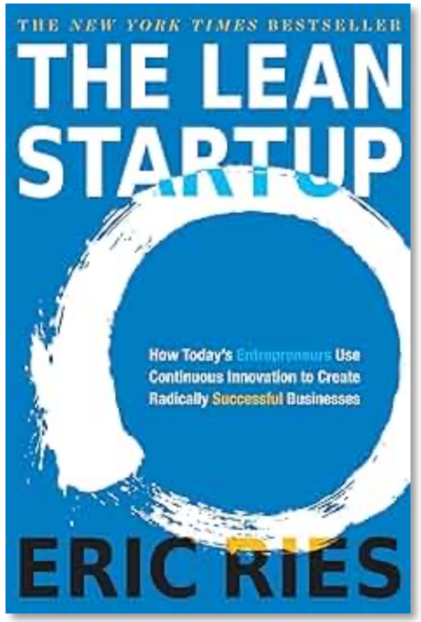
Alignment

~~Leadership Challenges~~	Start Up Challenges	Direction
Teamwork Challenges	Maps and Measures	Org Challenges
Driver Trees	Paths and Practice	Coordinated Strategy

Part 2
Start-up challenges

Start-up drama

- Hire good people, get out their way
- Optimize for adaptability
- Validated Learning


THE NEW YORK TIMES BESTSELLER
THE LEAN STARTUP
How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses
ERIC RIES

Start-up drama

- Inconsistent
- Unpredictable
- Fragile

Start-up Drama - Tension

- Fast
- Autonomy
- Team
- Top talent

- Reckless
- Divergent
- Mobbing/Burn out
- Workforce volatility

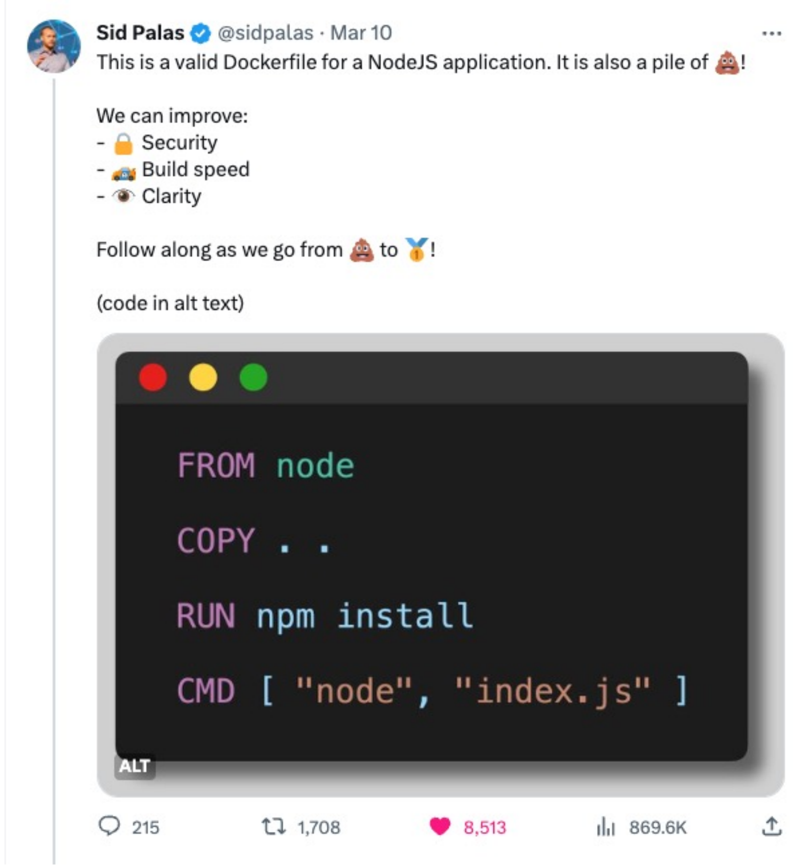
Finding a way

- Direction
- Map
- Path

~~Leadership Challenges~~	~~Start Up Challenges~~	Direction
Teamwork Challenges	Maps and Measures	Org Challenges
Driver Trees	Paths and Practice	Coordinated Strategy

Direction

Direction – Reference Example



Sid Palas @sidpalas · Mar 10

This is a valid Dockerfile for a NodeJS application. It is also a pile of 💩!

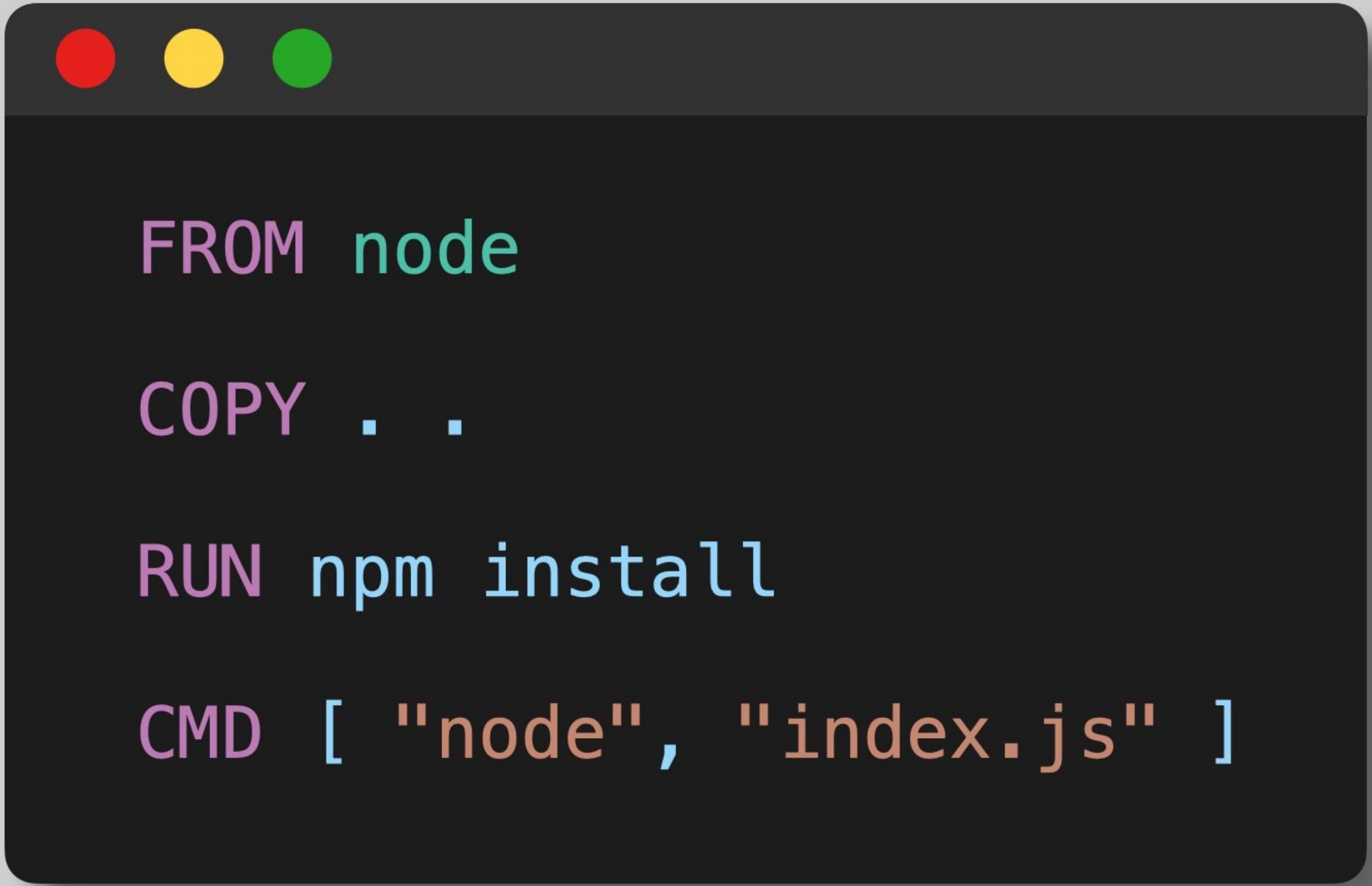
We can improve:
- 🔒 Security
- 🏎️ Build speed
- 👁️ Clarity

Follow along as we go from 💩 to 🥇!

(code in alt text)

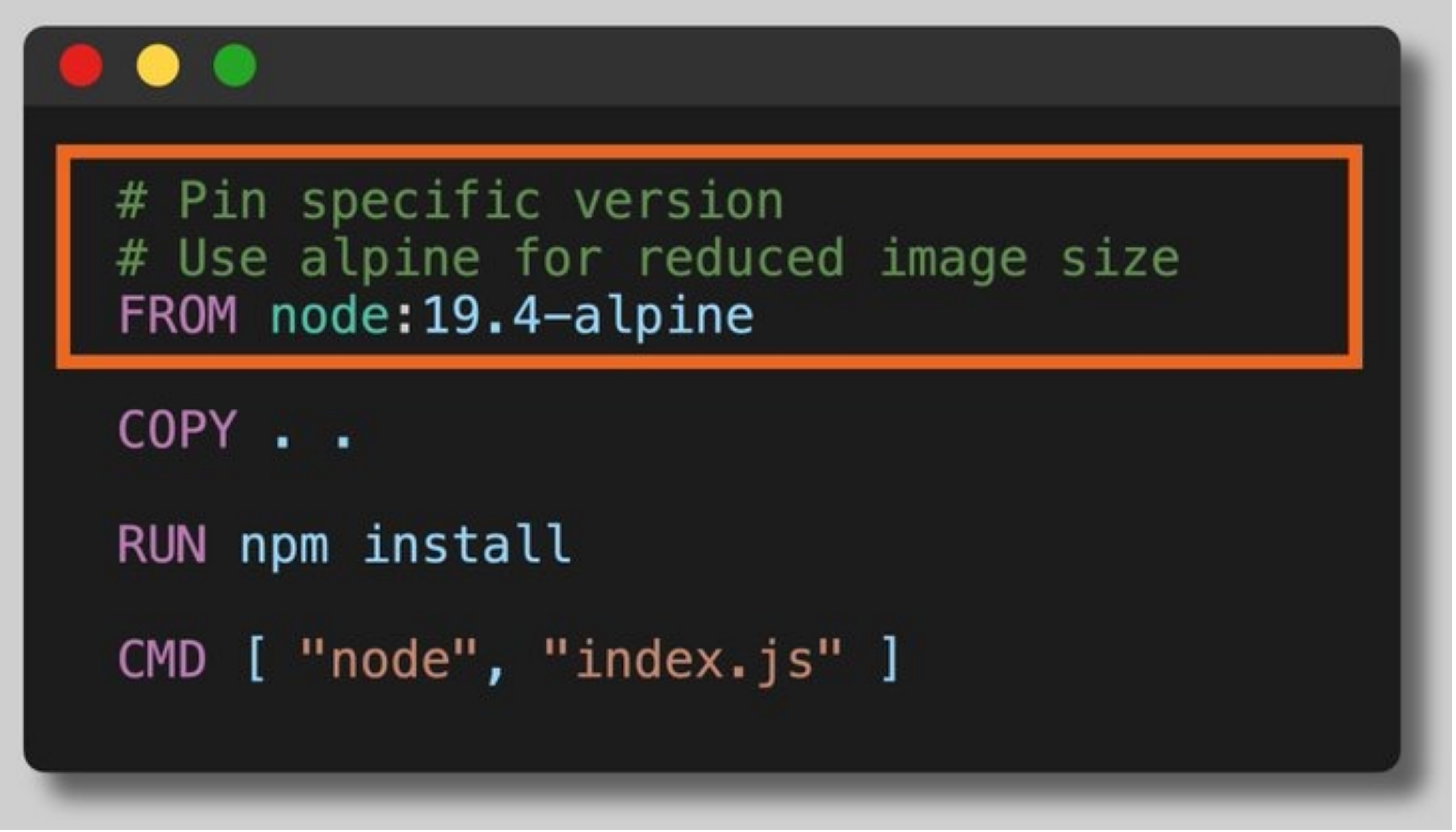
```
FROM node

COPY . .

RUN npm install

CMD [ "node", "index.js" ]
```

215 1,708 8,513 869.6K



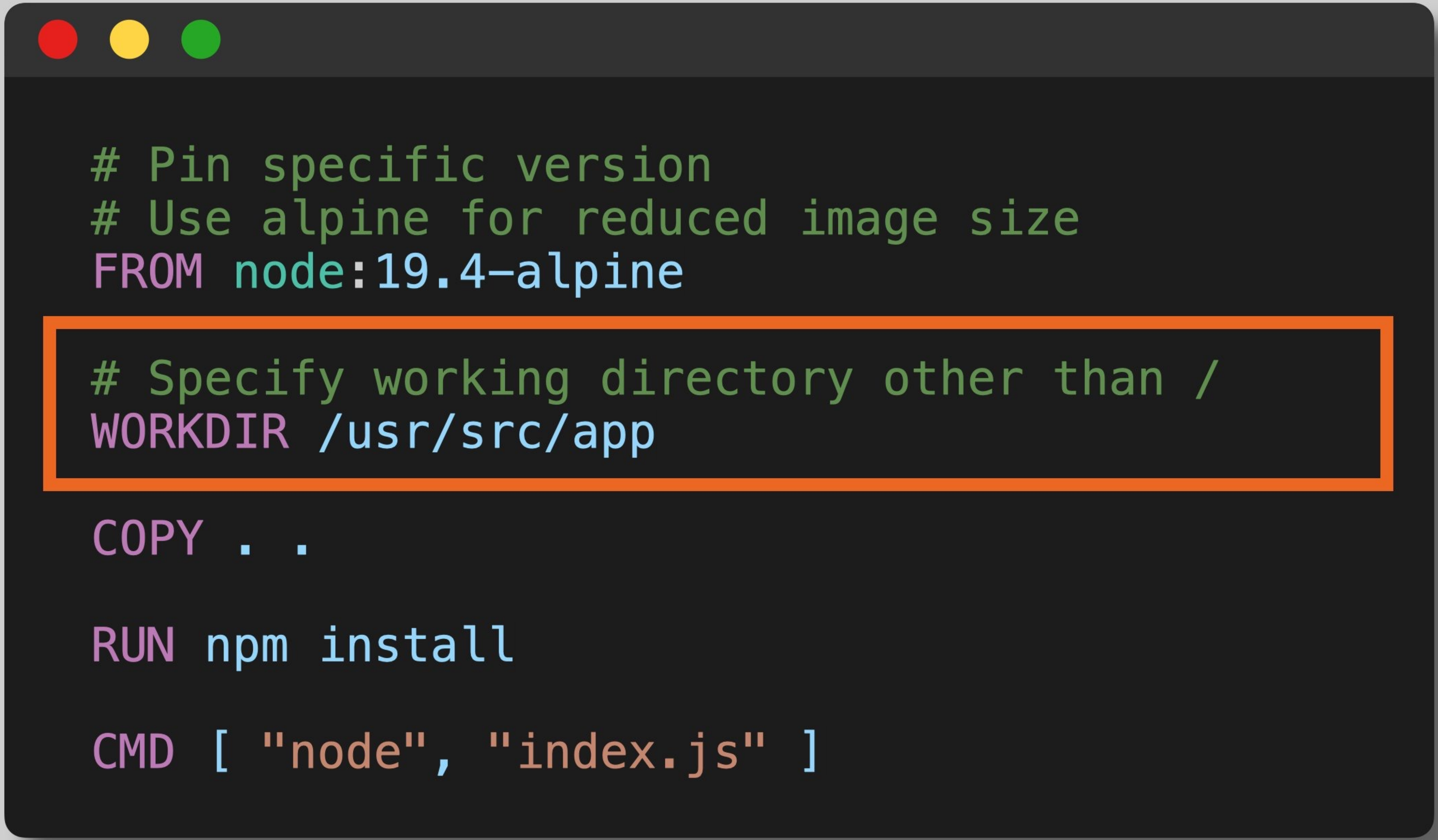
```
FROM node

COPY . .

RUN npm install

CMD [ "node", "index.js" ]
```

@SIDPALAS



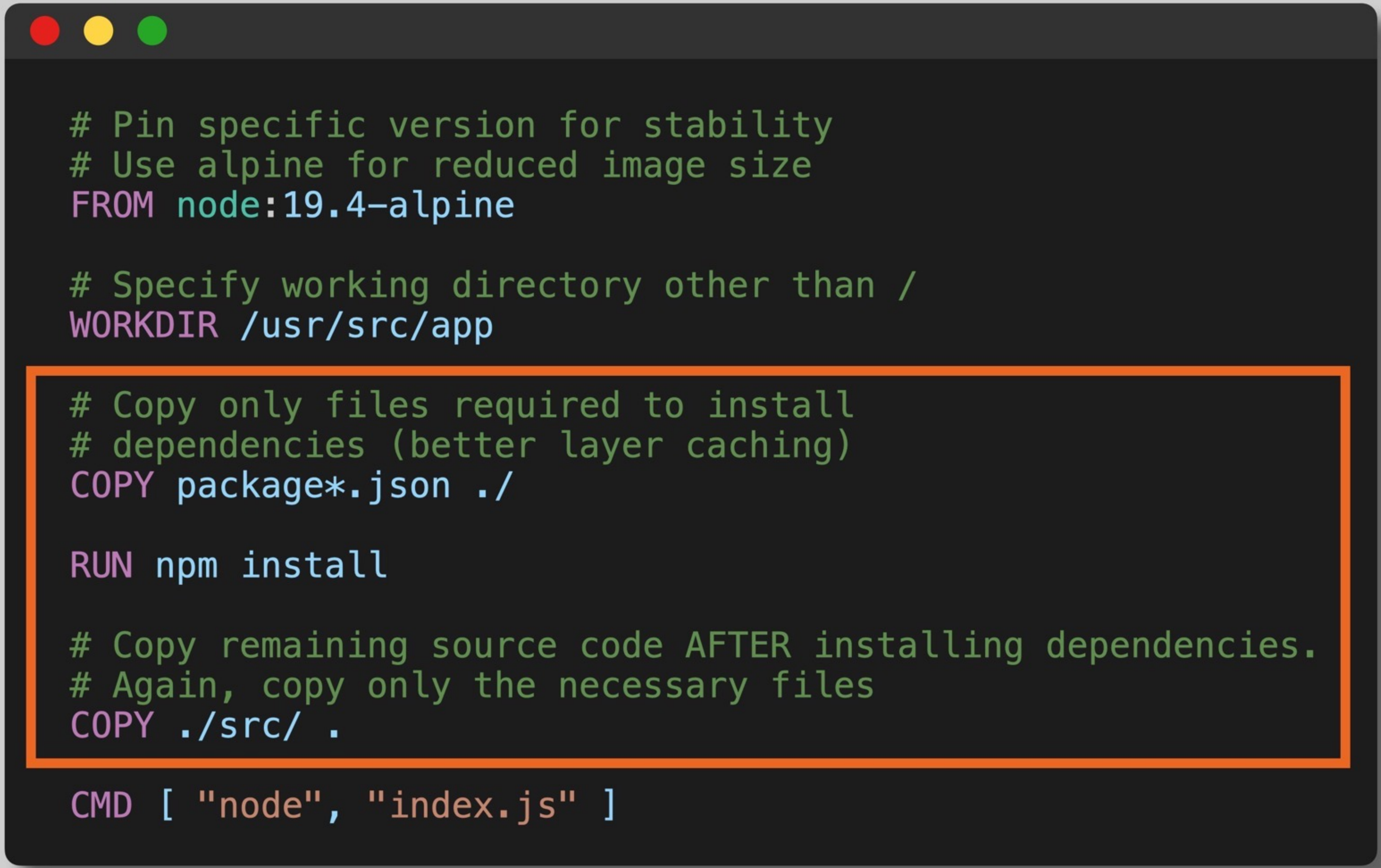
```
# Pin specific version
# Use alpine for reduced image size
FROM node:19.4-alpine

COPY . .

RUN npm install

CMD [ "node", "index.js" ]
```

@SIDPALAS



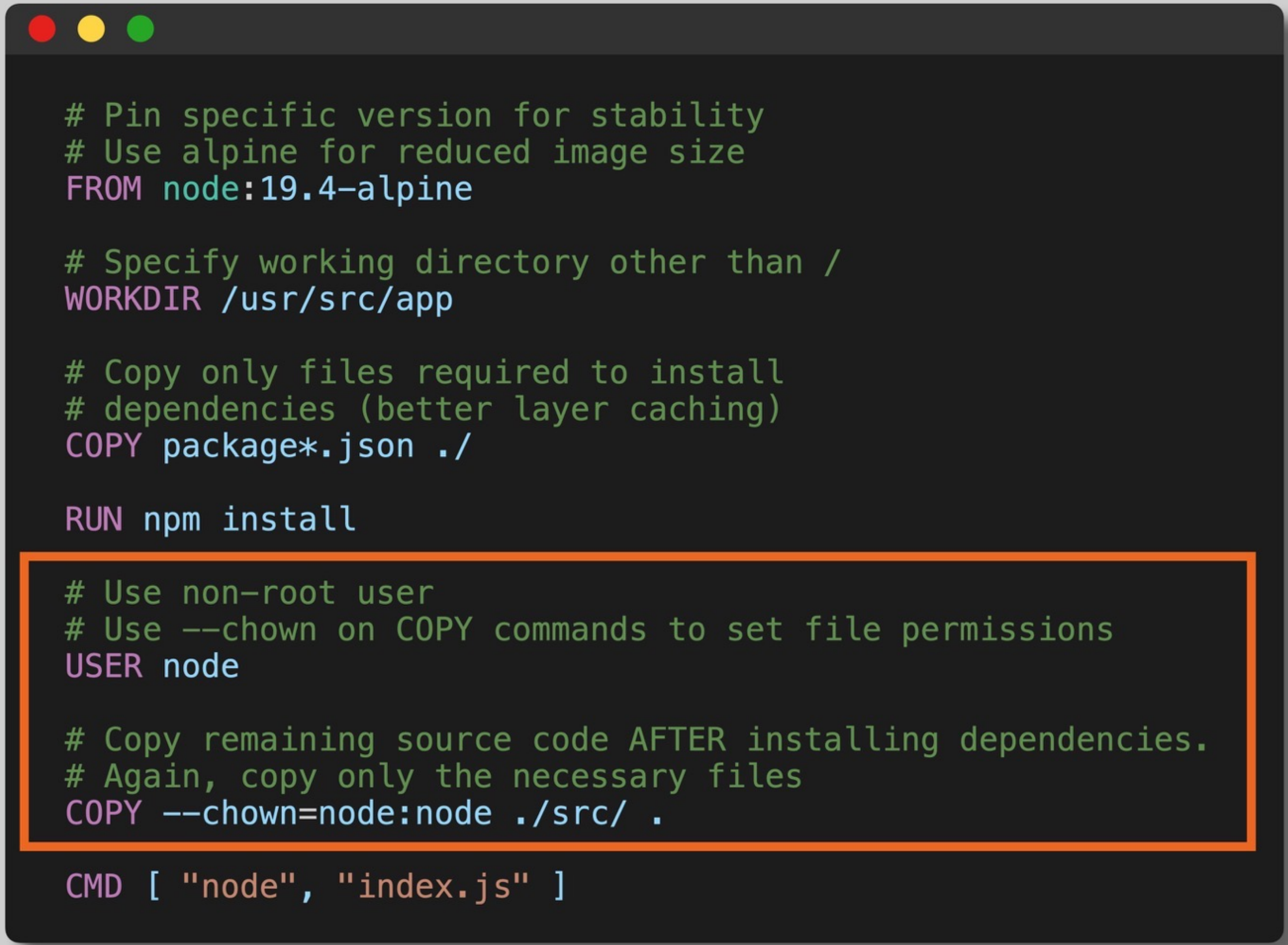
```
# Pin specific version
# Use alpine for reduced image size
FROM node:19.4-alpine

# Specify working directory other than /
WORKDIR /usr/src/app

COPY . .

RUN npm install

CMD [ "node", "index.js" ]
```

@SIDPALAS



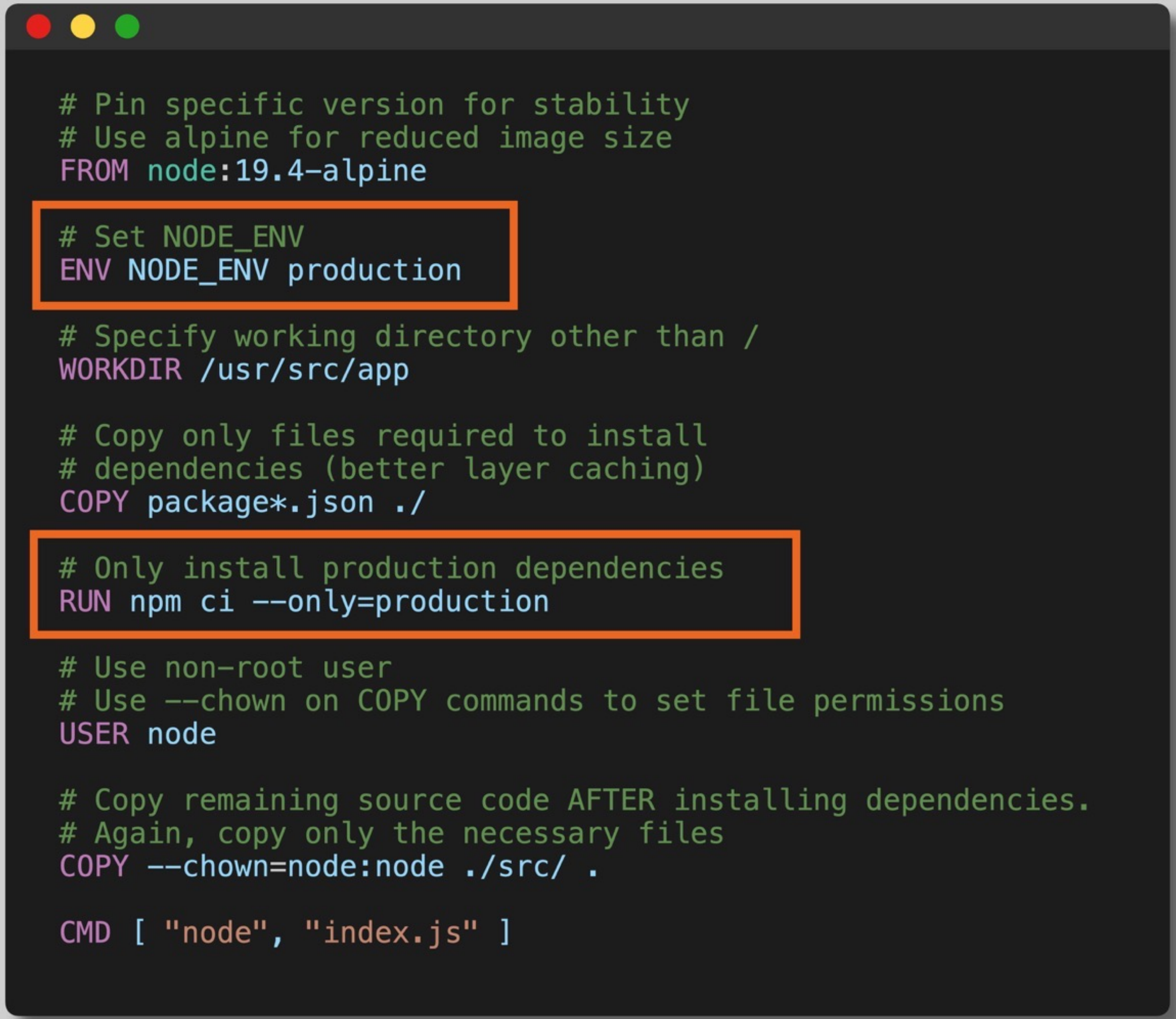
```
# Pin specific version for stability
# Use alpine for reduced image size
FROM node:19.4-alpine

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

RUN npm install

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY ./src/ .

CMD [ "node", "index.js" ]
```

@SIDPALAS



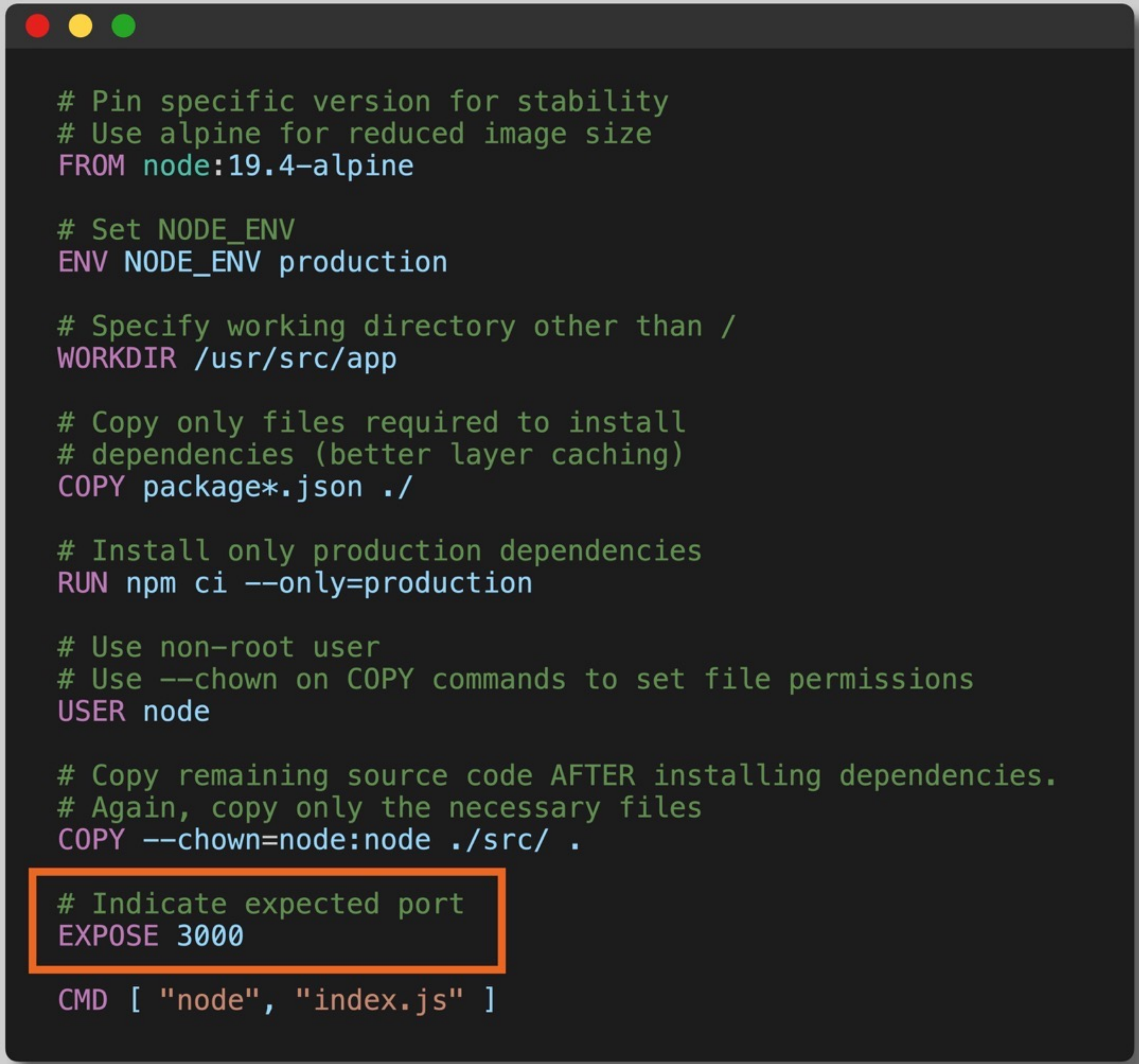
```
# Pin specific version for stability
# Use alpine for reduced image size
FROM node:19.4-alpine

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

RUN npm install

# Use non-root user
# Use --chown on COPY commands to set file permissions
USER node

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY --chown=node:node ./src/ .

CMD [ "node", "index.js" ]
```

@SIDPALAS



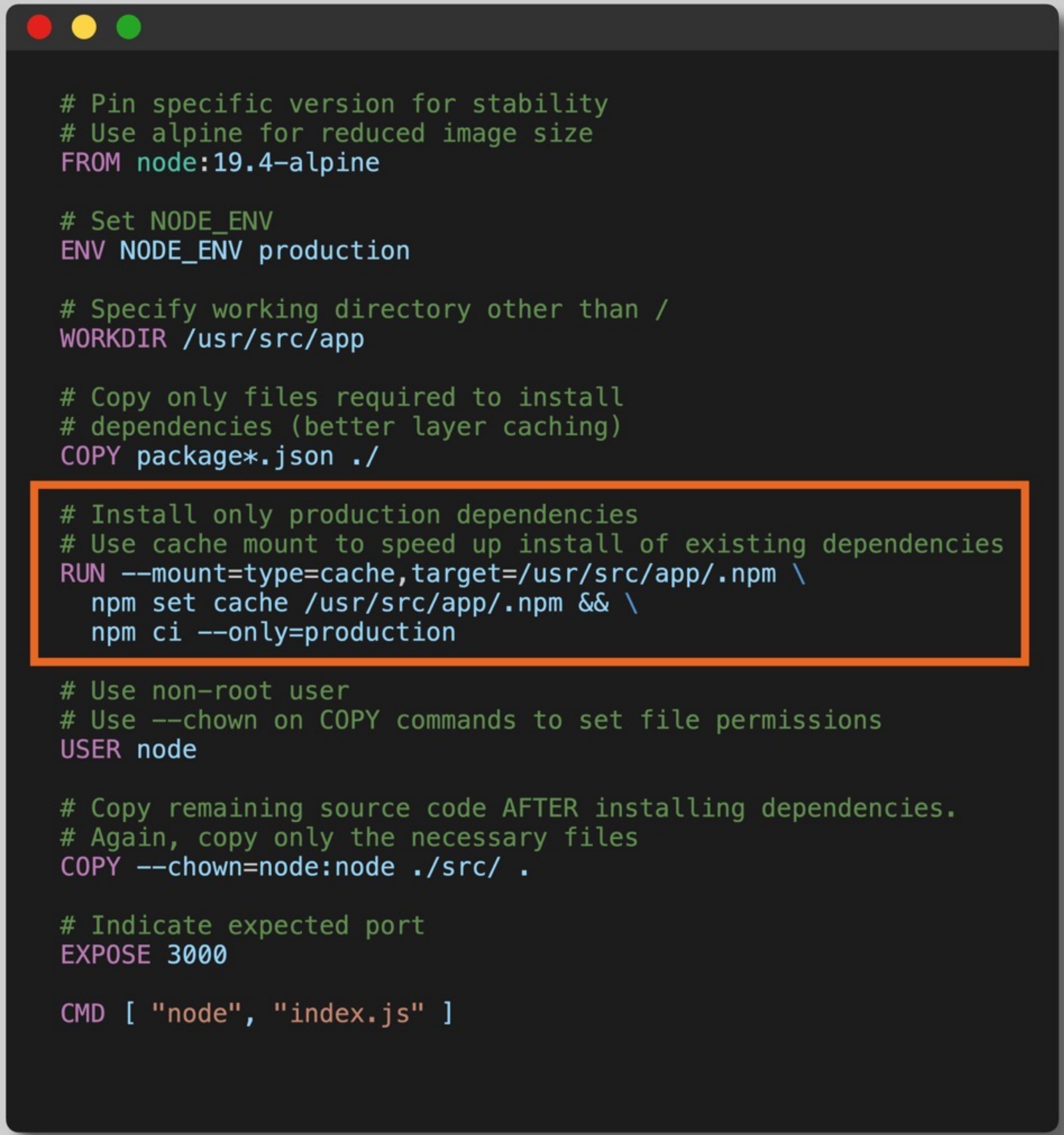
```
# Pin specific version for stability
# Use alpine for reduced image size
FROM node:19.4-alpine

# Set NODE_ENV
ENV NODE_ENV production

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

# Only install production dependencies
RUN npm ci --only=production

# Use non-root user
# Use --chown on COPY commands to set file permissions
USER node

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY --chown=node:node ./src/ .

CMD [ "node", "index.js" ]
```

@SIDPALAS



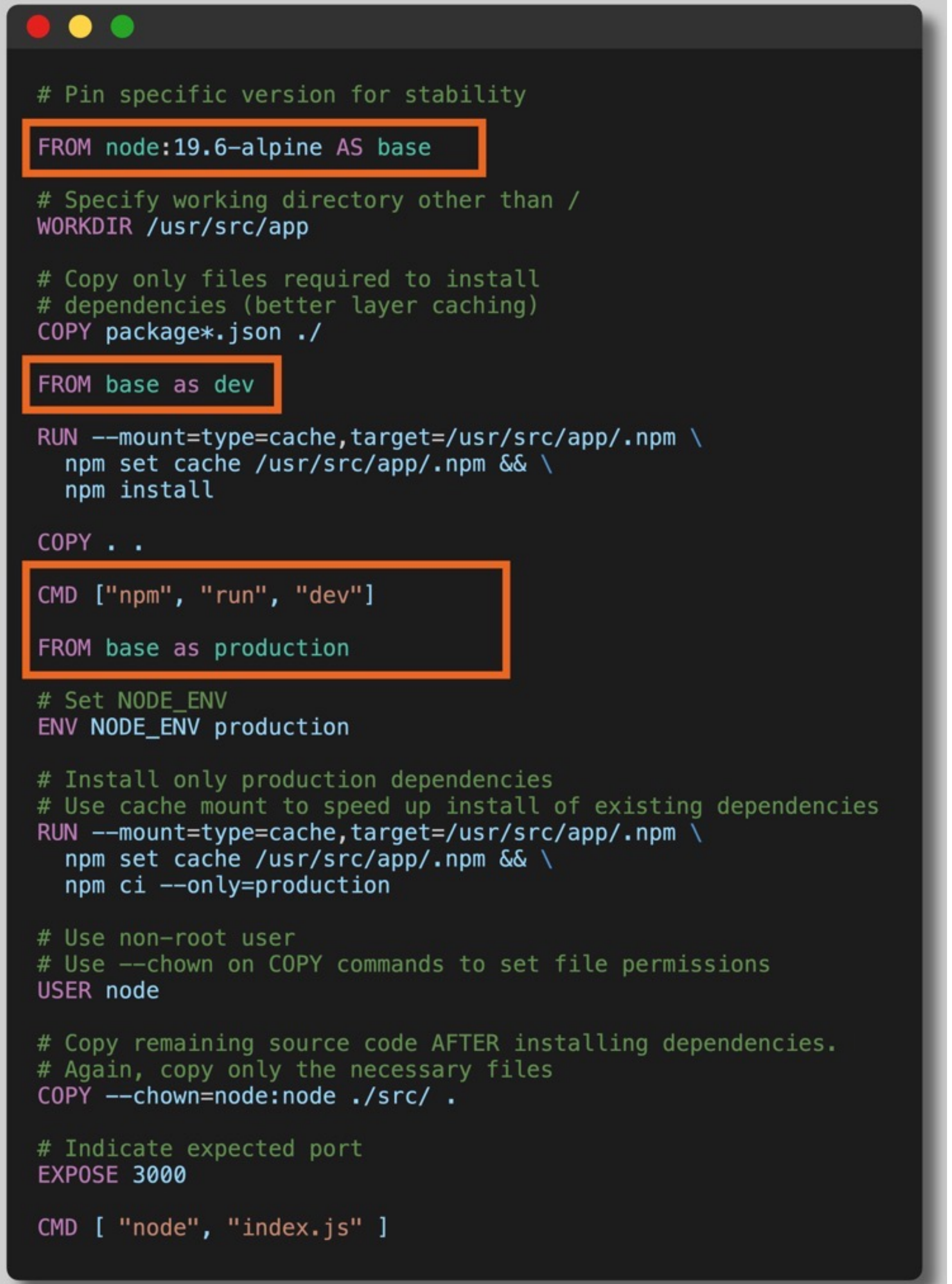
```dockerfile
# Pin specific version for stability
# Use alpine for reduced image size
FROM node:19.4-alpine

# Set NODE_ENV
ENV NODE_ENV production

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

# Install only production dependencies
RUN npm ci --only=production

# Use non-root user
# Use --chown on COPY commands to set file permissions
USER node

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY --chown=node:node ./src/ .

# Indicate expected port
EXPOSE 3000

CMD [ "node", "index.js" ]
```

@SIDPALAS



```dockerfile
# Pin specific version for stability
# Use alpine for reduced image size
FROM node:19.4-alpine

# Set NODE_ENV
ENV NODE_ENV production

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

# Install only production dependencies
# Use cache mount to speed up install of existing dependencies
RUN --mount=type=cache,target=/usr/src/app/.npm \
  npm set cache /usr/src/app/.npm && \
  npm ci --only=production

# Use non-root user
# Use --chown on COPY commands to set file permissions
USER node

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY --chown=node:node ./src/ .

# Indicate expected port
EXPOSE 3000

CMD [ "node", "index.js" ]
```

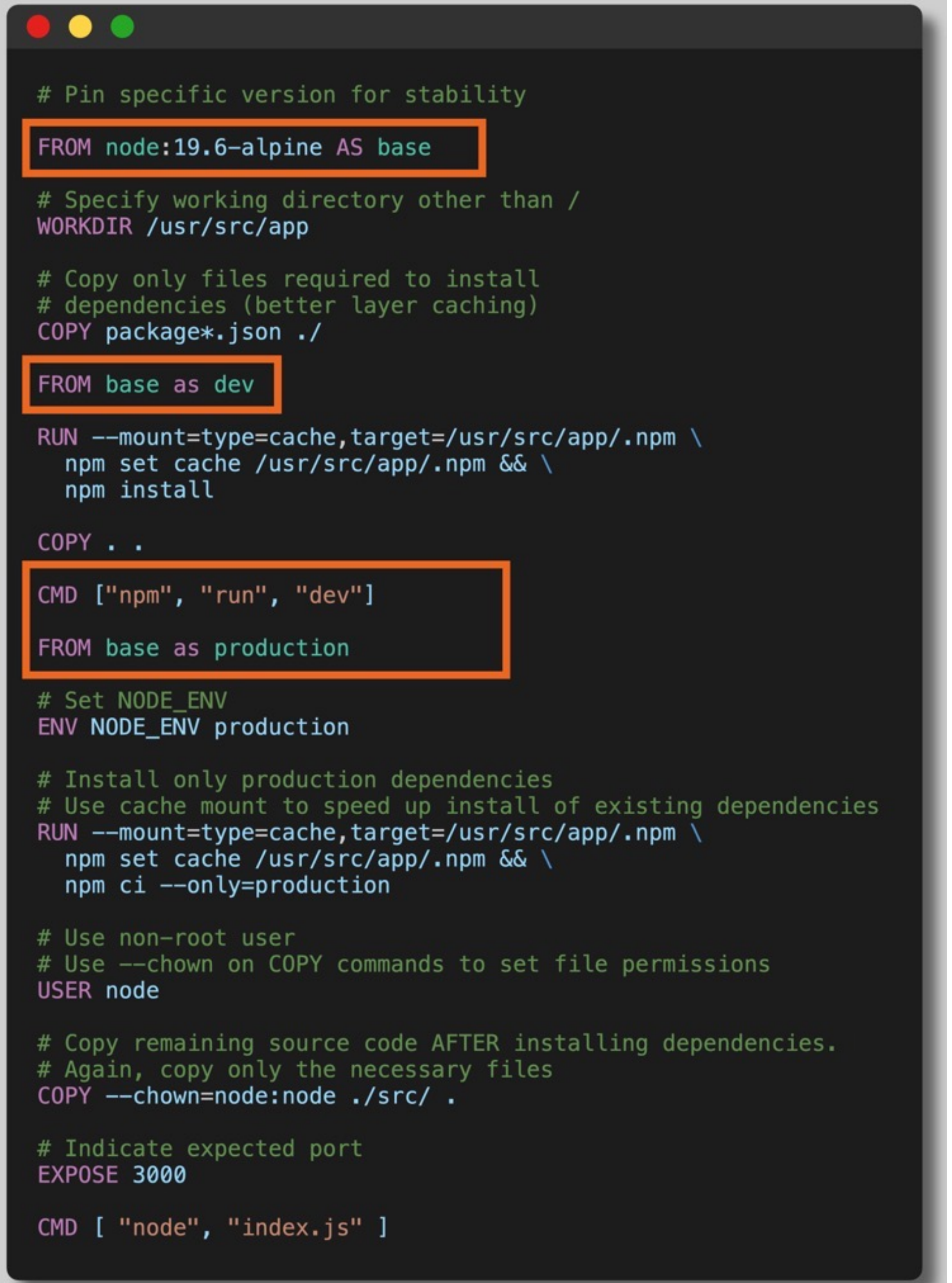
@SIDPALAS



```dockerfile
# Pin specific version for stability
FROM node:19.6-alpine AS base

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

FROM base as dev

RUN --mount=type=cache,target=/usr/src/app/.npm \
  npm set cache /usr/src/app/.npm && \
  npm install

COPY . .

CMD ["npm", "run", "dev"]

FROM base as production

# Set NODE_ENV
ENV NODE_ENV production

# Install only production dependencies
# Use cache mount to speed up install of existing dependencies
RUN --mount=type=cache,target=/usr/src/app/.npm \
  npm set cache /usr/src/app/.npm && \
  npm ci --only=production

# Use non-root user
# Use --chown on COPY commands to set file permissions
USER node

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY --chown=node:node ./src/ .

# Indicate expected port
EXPOSE 3000

CMD [ "node", "index.js" ]
```

Search: “sidpalas dockerfile”

@SIDPALAS



```
# Pin specific version for stability
FROM node:19.6-alpine AS base

# Specify working directory other than /
WORKDIR /usr/src/app

# Copy only files required to install
# dependencies (better layer caching)
COPY package*.json ./

FROM base as dev

RUN --mount=type=cache,target=/usr/src/app/.npm \
  npm set cache /usr/src/app/.npm && \
  npm install

COPY . .

CMD ["npm", "run", "dev"]

FROM base as production

# Set NODE_ENV
ENV NODE_ENV production

# Install only production dependencies
# Use cache mount to speed up install of existing dependencies
RUN --mount=type=cache,target=/usr/src/app/.npm \
  npm set cache /usr/src/app/.npm && \
  npm ci --only=production

# Use non-root user
# Use --chown on COPY commands to set file permissions
USER node

# Copy remaining source code AFTER installing dependencies.
# Again, copy only the necessary files
COPY --chown=node:node ./src/ .

# Indicate expected port
EXPOSE 3000

CMD [ "node", "index.js" ]
```

Search: “sidpalas dockerfile”

@SIDPALAS

Direction – Performance Expectations

- Same quality, but different speeds
- → overload the stars
- → can’t make plans

Direction – Performance Expectations

- Provide a reference of what good looks like
- Set time expectations to perform

Direction – Business Outcomes

Not Business Outcomes:

- “Deploy Kubernetes cluster”
- “Update WebAPI”
- “Partition topic”

Direction – Business Outcomes

Estimation:

- Are we estimating on the same thing?
- Who designed the “thing”?
- Why is ~~effort~~ time the only negotiable

Direction – Business Outcomes

Estimation:

- Assumes the solution
- Negotiates on the wrong thing
- Spends cognitive energy on the wrong thing

Direction – Business Outcomes

Business outcomes:

- Reduce closures from failed checks
  - …

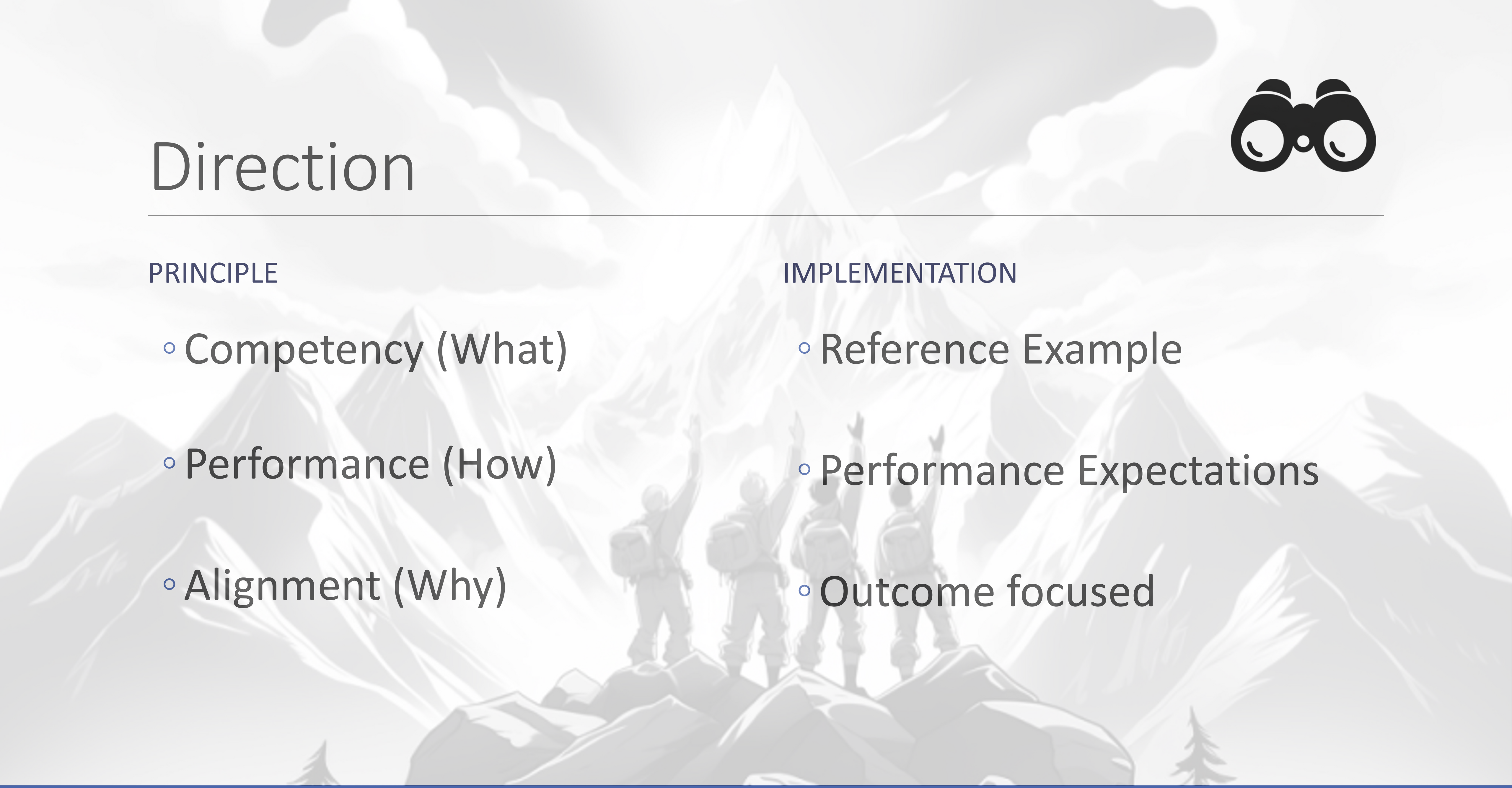
Direction – Business Outcomes

Business outcomes:

- Reduce closures from failed checks
  - by notifying sales of accounts under credit check

- Increase C2L
  - by reducing page load times
- Increase registration rate
  - by reducing user interaction count

Direction



PRINCIPLE

- Competency (What)
- Performance (How)
- Alignment (Why)

IMPLEMENTATION

- Reference Example
- Performance Expectations
- Outcome focused

~~Leadership Challenges~~	~~Start Up Challenges~~	~~Direction~~
Teamwork Challenges	Maps and Measures	Org Challenges
Driver Trees	Paths and Practice	Coordinated Strategy

Part 3
Teamwork Challenges

Teamwork Challenges – People

- Who does what and when?
- Where can I innovate?
- How do I grow and get promoted?

Teamwork Challenges - Systems

- What should this system do?
- Scrappy not Crappy?
- Gold plating?
- Is Tech Debt bad?

~~Leadership Challenges~~	~~Start Up Challenges~~	~~Direction~~
~~Teamwork Challenges~~	**Maps and Measures**	Org Challenges
Driver Trees	Paths and Practice	Coordinated Strategy

Maps and Measures

1. Roles and Responsibilities
2. Quality of Service
3. Measurable outcomes

Maps - Roles & Responsibilities

- Who is accountable for missed deadlines and outages
- What are your steps for promotion
- How do you ensure constant business delivery

Maps - Roles & Responsibilities

- Define skill sets
- For each role, define
  - Expected competency
  - Organizational scope it applies to
- Create a Competency Matrix

Maps - Roles & Responsibilities

	Title 1	Title 2	Title n
Plan it			
Build it			
Ship it			
Run it			

Maps - Roles & Responsibilities

	Junior	Senior	Principal
Plan it	Attends planning	Plans a team's project	Guide multiple Seniors
	Can identify business goals	Aligns plans to business goals	Ensures team's plans are aligned
Build it	Contributes to development	Accountable for development for team	Defines development practices across teams
	Can identify Tech Debt	Documents tech debt	Approves tech debt
Ship it	Deploys code	Releases features	Defines Release process
	Keeps build green	Keeps builds fast	Coaches CI/CD
Run it	Can identify the team SLOs	Ensures team SLOs are measured	Defines SLOs for multiple teams
	Attends Incident Response training	First responder for team	Incident Commander for multiple teams

Maps - Roles & Responsibilities

	Junior	Senior	Principal
Plan it	Attends planning	Plans a team's project	Guide multiple Seniors
	Can identify business goals	Aligns plans to business goals	Ensures team's plans are aligned
Build it	Contributes to development	Accountable for development for team	Defines development practices across teams
	Can identify Tech Debt	Documents tech debt	Approves tech debt
Ship it	Deploys code	Releases features	Defines Release process
	Keeps build green	Keeps builds fast	Coaches CI/CD
Run it	Can identify the team SLOs	Ensures team SLOs are measured	Defines SLOs for multiple teams
	Attends Incident Response training	First responder for team	Incident Commander for multiple teams

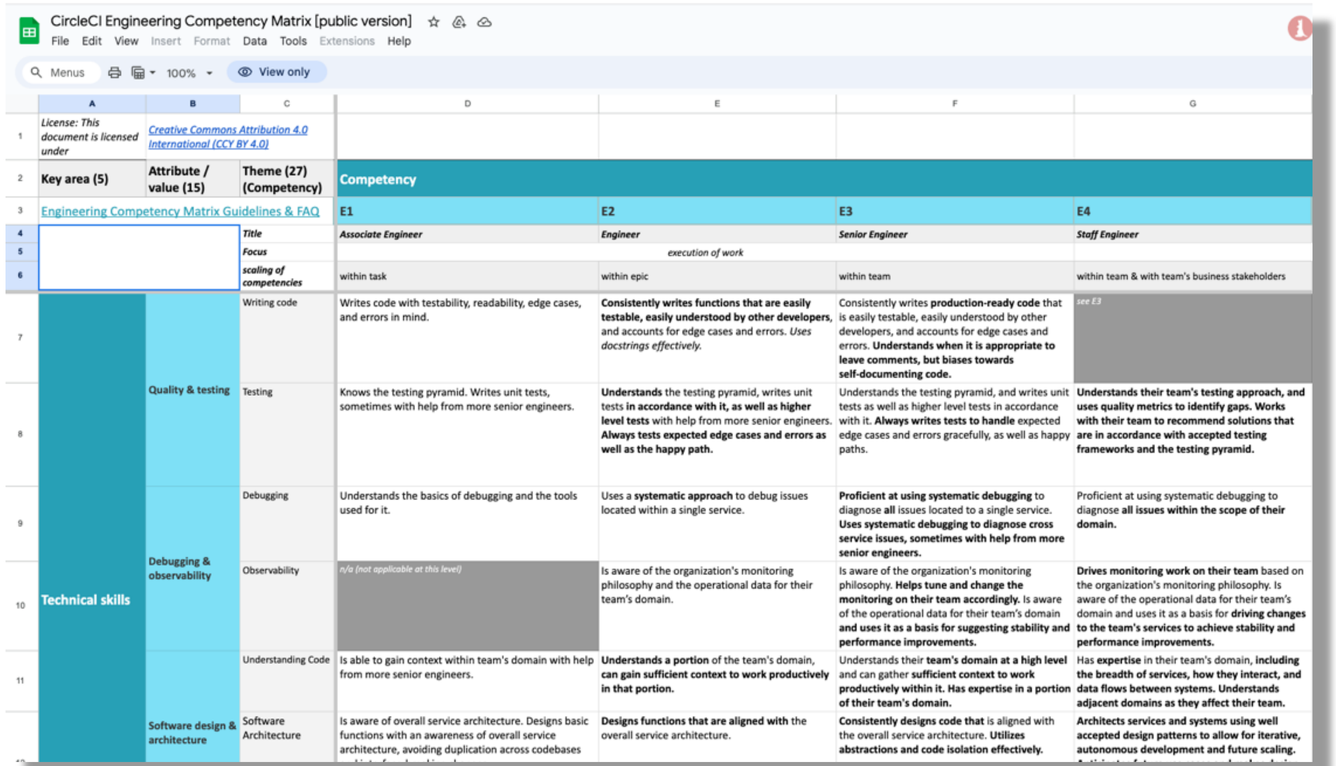
Maps - Roles & Responsibilities

	Junior	Senior	Principal
Plan it			
Build it			
Ship it			
Run it			

Maps - Roles & Responsibilities

Search: “Circle CI Competency Matrix”

Search: “SFIA” or “CMM”



CircleCI Engineering Competency Matrix [public version]

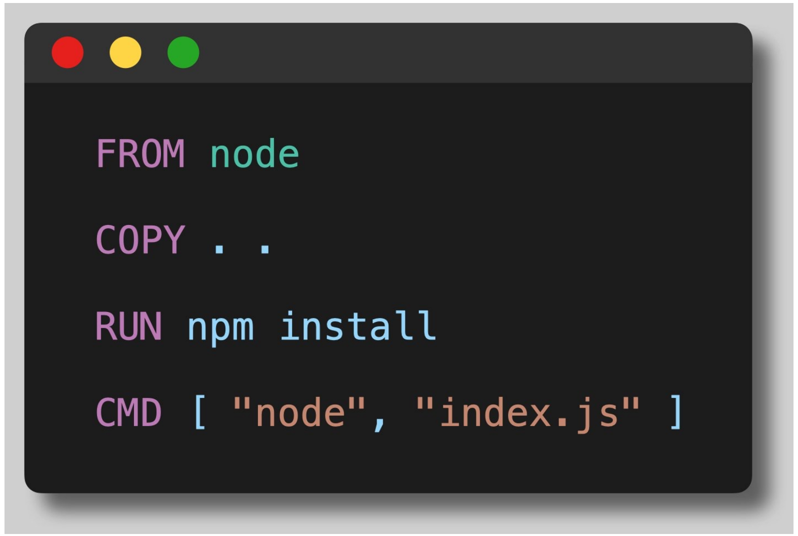
A	B	C	D	E	F	G
License: This document is licensed under	*Creative Commons Attribution 4.0 International (CCY BY 4.0)*					
Key area (5)	**Attribute / value (15)**	**Theme (27) (Competency)**	**Competency**			
Engineering Competency Matrix Guidelines & FAQ			**E1**	**E2**	**E3**	**E4**
		Title	***Associate Engineer***	***Engineer***	***Senior Engineer***	***Staff Engineer***
		Focus		*execution of work*		
		scaling of competencies	within task	within epic	within team	within team & with team's business stakeholders
Technical skills	**Quality & testing**	Writing code	Writes code with testability, readability, edge cases, and errors in mind.	**Consistently writes functions that are easily testable, easily understood by other developers,** and accounts for edge cases and errors. *Uses docstrings effectively.*	Consistently writes **production-ready code** that is easily testable, easily understood by other developers, and accounts for edge cases and errors. **Understands when it is appropriate to leave comments, but biases towards self-documenting code.**	*see E3*
		Testing	Knows the testing pyramid. Writes unit tests, sometimes with help from more senior engineers.	**Understands** the testing pyramid, writes unit tests **in accordance with it, as well as higher level tests** with help from more senior engineers. **Always tests expected edge cases and errors as well as the happy path.**	Understands the testing pyramid, and writes unit tests as well as higher level tests in accordance with it. **Always writes tests to handle** expected edge cases and errors gracefully, as well as happy paths.	**Understands their team's testing approach, and uses quality metrics to identify gaps. Works with their team to recommend solutions that are in accordance with accepted testing frameworks and the testing pyramid.**
	Debugging & observability	Debugging	Understands the basics of debugging and the tools used for it.	Uses a **systematic approach** to debug issues located within a single service.	**Proficient at using systematic debugging** to diagnose **all** issues located to a single service. **Uses systematic debugging to diagnose cross service issues, sometimes with help from more senior engineers.**	Proficient at using systematic debugging to diagnose **all issues within the scope of their domain.**
		Observability	*n/a (not applicable at this level)*	Is aware of the organization's monitoring philosophy and the operational data for their team's domain.	Is aware of the organization's monitoring philosophy. **Helps tune and change the monitoring on their team accordingly.** Is aware of the operational data for their team's domain **and uses it as a basis for suggesting stability and performance improvements.**	**Drives monitoring work on their team** based on the organization's monitoring philosophy. Is aware of the operational data for their team's domain and uses it as a basis for **driving changes to the team's services to achieve stability and performance improvements.**
	Software design & architecture	Understanding Code	Is able to gain context within team's domain with help from more senior engineers.	**Understands a portion** of the team's domain, **can gain sufficient context to work productively in that portion.**	Understands their **team's domain at a high level** and can gather **sufficient context to work productively within it. Has expertise in a portion of their team's domain.**	Has **expertise** in their team's domain, **including the breadth of services, how they interact, and data flows between systems. Understands adjacent domains as they affect their team.**
		Software Architecture	Is aware of overall service architecture. Designs basic functions with an awareness of overall service architecture, avoiding duplication across codebases	**Designs functions that are aligned with** the overall service architecture.	**Consistently designs code that** is aligned with the overall service architecture. **Utilizes abstractions and code isolation effectively.**	**Architects services and systems using well accepted design patterns to allow for iterative, autonomous development and future scaling.**

Maps - System Maturity

- What is expected of my system?
- Should I be doing
  - Chaos Engineering or Reactive Programming?
  - Distributed Tracing or Security Patches?
  - Disaster Recover Planning or Data Mesh?

Maps - System Maturity

1. *[Step 1]*
2. *[Step 2]*
3. Provide a reference of what good looks like



```
FROM node

COPY . .

RUN npm install

CMD [ "node", "index.js" ]
```

Maps - System Maturity

1. Define your technology choices
2. Limit your technology choices
3. Provide a reference of what good looks like

Maps - System Maturity

- Define your technology choices
- Limit your technology choices
- Provide a reference of what good looks like

Maps - System Maturity

	1-Adhoc	2-Consistent	3-Systematized	4-Strategic
Plan it				
Build it				
Ship it				
Run it				

Maps - System Maturity

	1-Adhoc	2-Consistent	3-Systematized	4-Strategic
Plan it	Work is communicated	Work is planned	Work is prioritized	Priorities align with other teams
	System workload is known	Planned workloads	Workloads are cataloged	Workloads are elastic
Build it	Documented coding standards	Reference Example	Coding standards enforced	ADRs
	Trunk Based	Pinned versions	SemVer Releases	Automated Release
Ship it	Releases are communicated	Releases are gated by independent QC	Rollback/forward is planned	Rollback is automated
	Infra documented	Infra is tagged	Infra as Code	Policy as Code
Run it	Logging	Traces & Metrics	SLIs and SLOs	Error Budgets
	On call policy	On call roster	Incident Response	DR trained

Maps - Measurable Outcomes

Business outcomes:

- Reduce closures from failed checks **from 10 to 2 per month**
  - …

Maps - Measurable Outcomes

Business outcomes:

- Reduce closures from failed checks **from 10 to 2 per month**
  - by notifying sales of accounts under credit check **within 5min**

- Increase C2L **from 50% to 80%**
  - by reducing page load times **to under 2s**
- Increase registration rate **from 10% to 20%**
  - by reducing user interaction count **from 18 to <10**

Maps and Measures

PRINCIPLE

- Competency (What)
- Performance (How)
- Alignment (Why)

IMPLEMENTATION

- Competency Matrix
- Maturity Models
- Measurable outcomes

~~Leadership Challenges~~	~~Start Up Challenges~~	~~Direction~~
~~Teamwork Challenges~~	~~Maps and Measures~~	Org Challenges
Driver Trees	Paths and Practice	Coordinated Strategy

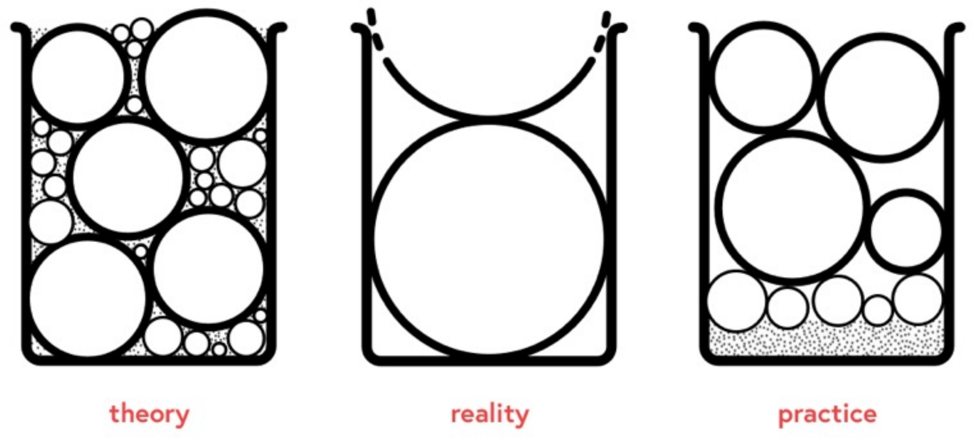
Part 3
Organizational Challenges

Organizational Challenges

- Big projects never get prioritized
- No time for “Weeding and Feeding”
- Performance expectations are hard to meet

Organizational Challenges
Strategic Failure

- KPIs, OKRs and “Rocks”
  - Solution in the outcome
  - Disconnection between work and goal
  - Metrics you can’t directly affect



Organizational Challenges
Strategic Failure

- **Solution is the outcome**
  - “Move to the cloud”
  - “Implement [Technology] by End of Year”
  - “Adopt [Process]”

Organizational Challenges Strategic Failure

- **Disconnect between work and goal**
  - Goal is “Reduce Cost-Per-Install”
  - Work is “Adopt Material design”

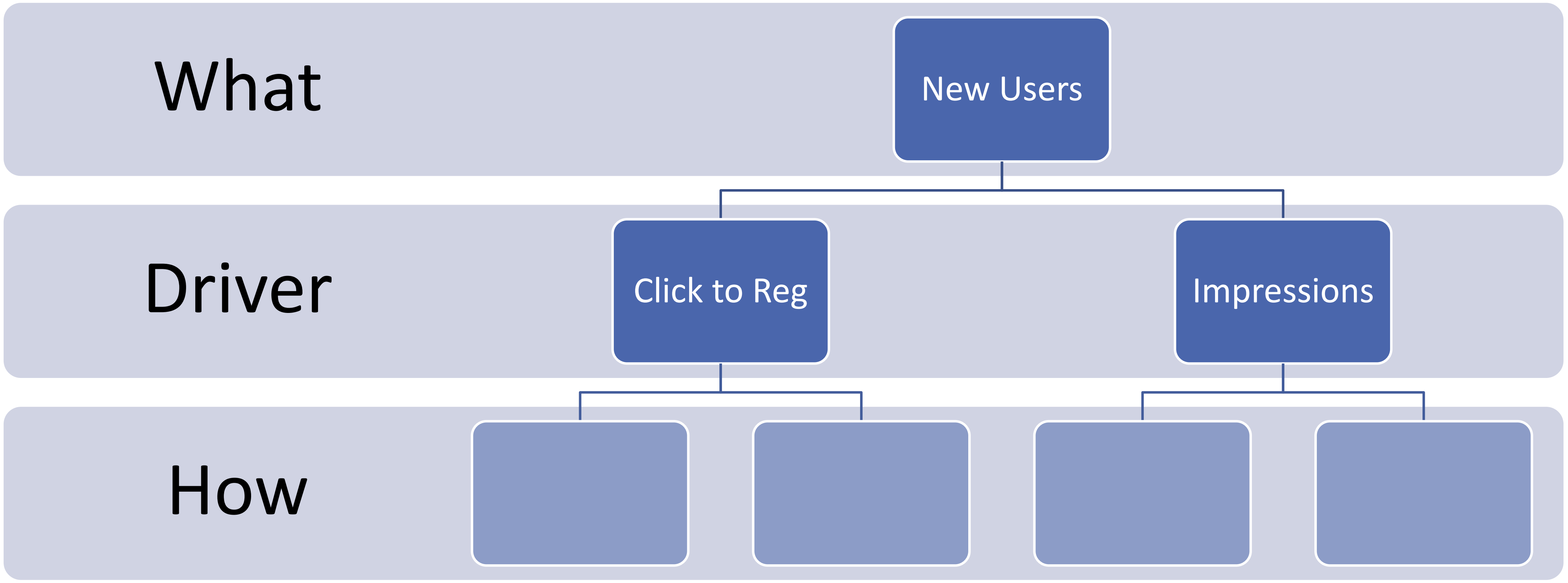
  - Goal is “Move to cloud”
  - Work is “Upgrade database”

Organizational Challenges
Strategic Failure

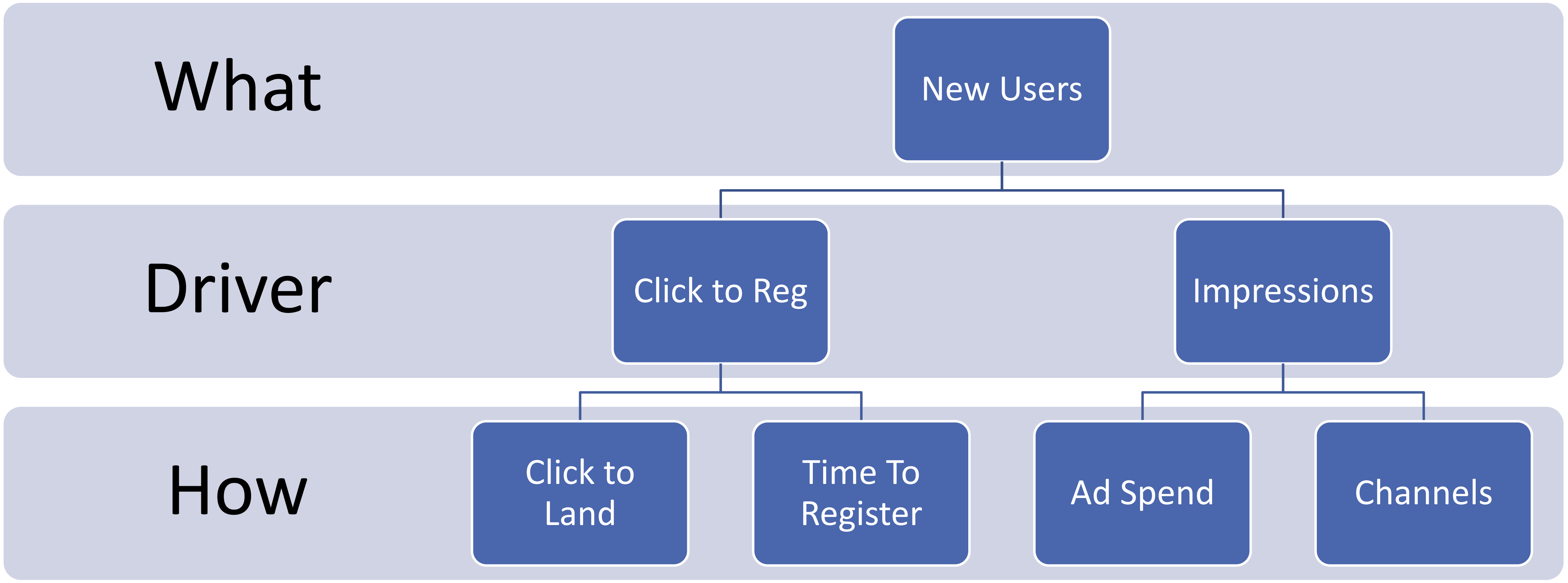
- **Metrics you can’t directly affect**
  - CPA (Cost per Acquisition) / CPI (Cost per Install)
  - ROAS (Return on Ad Spend)
  - Customer Retention Rate

~~Leadership Challenges~~	~~Start Up Challenges~~	~~Direction~~
~~Teamwork Challenges~~	~~Maps and Measures~~	~~Org Challenges~~
Driver Trees	Paths and Practice	Coordinated Strategy

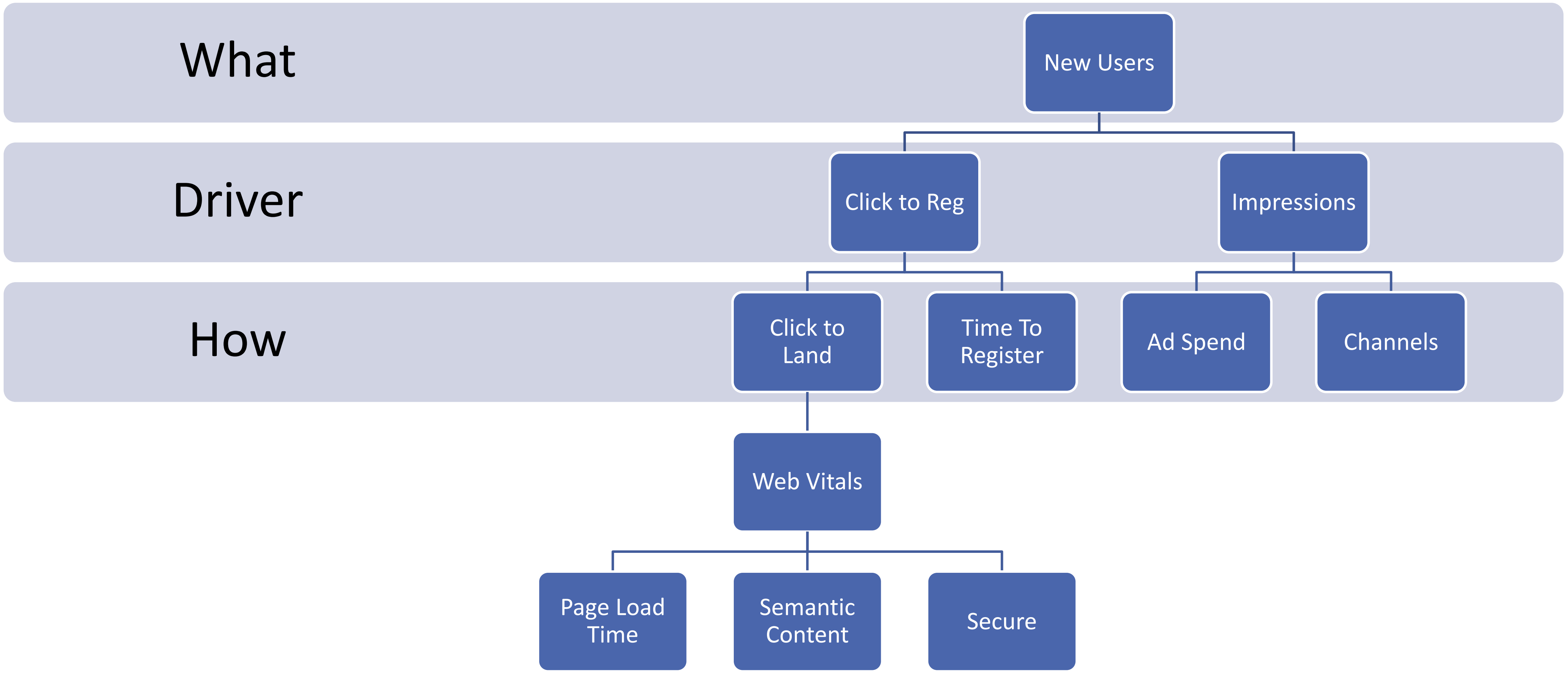
Driver Trees



Driver Trees



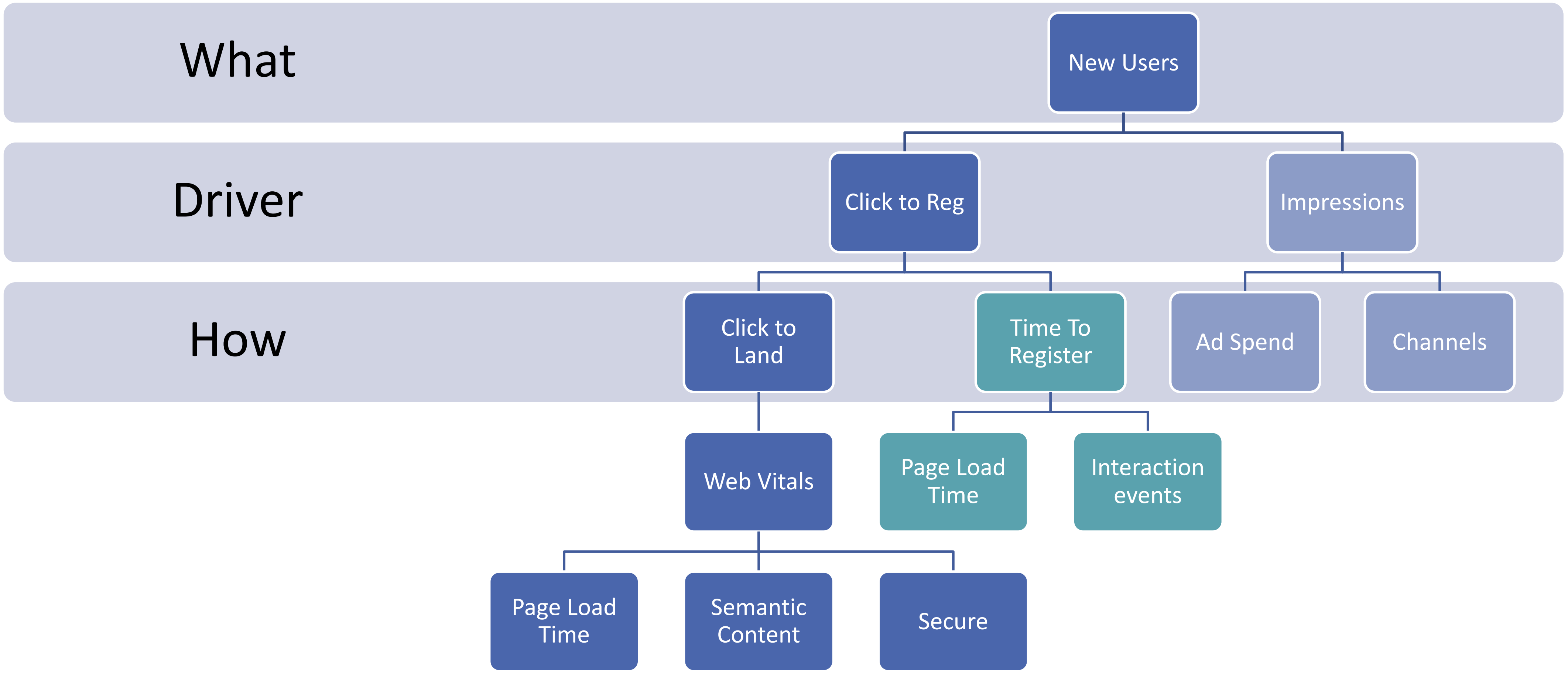
Driver Trees



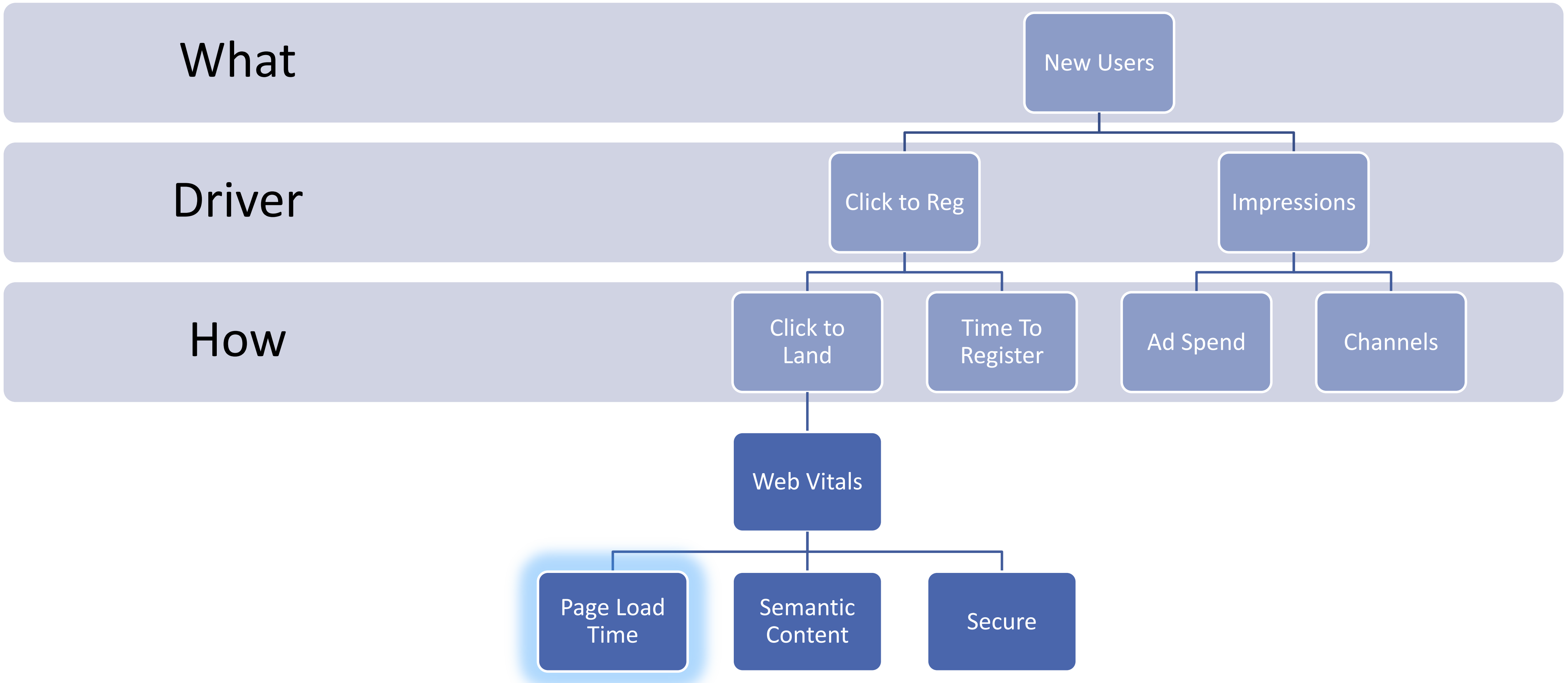
Driver Trees

- Driver trees
  - Find levers you can manipulate directly
  - Make bets on the efficacy on which lever to pull
  - Have a unified language

Driver Trees



Driver Trees



~~Leadership Challenges~~

~~Start Up Challenges~~

~~Direction~~

~~Teamwork Challenges~~

~~Maps and Measures~~

~~Org Challenges~~

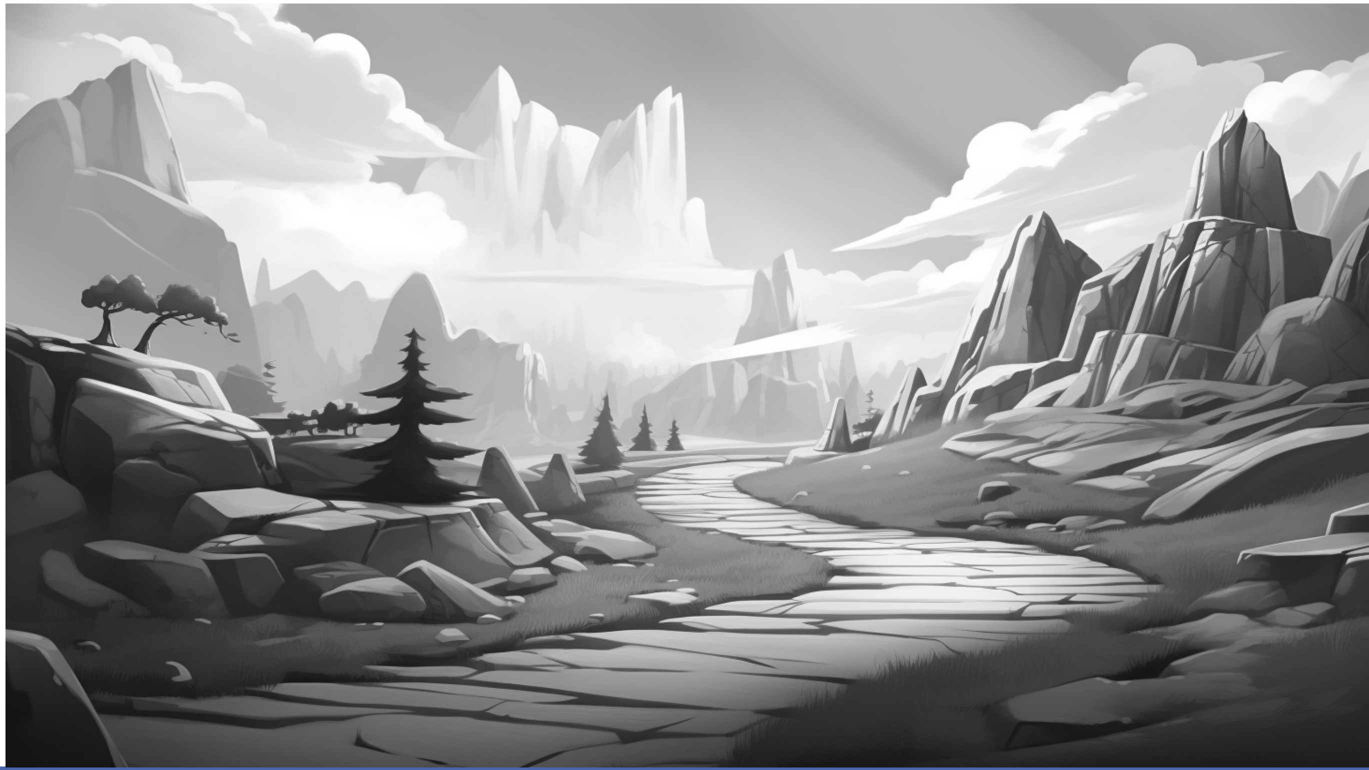
~~Driver Trees~~

Paths and Practice

Coordinated Strategy

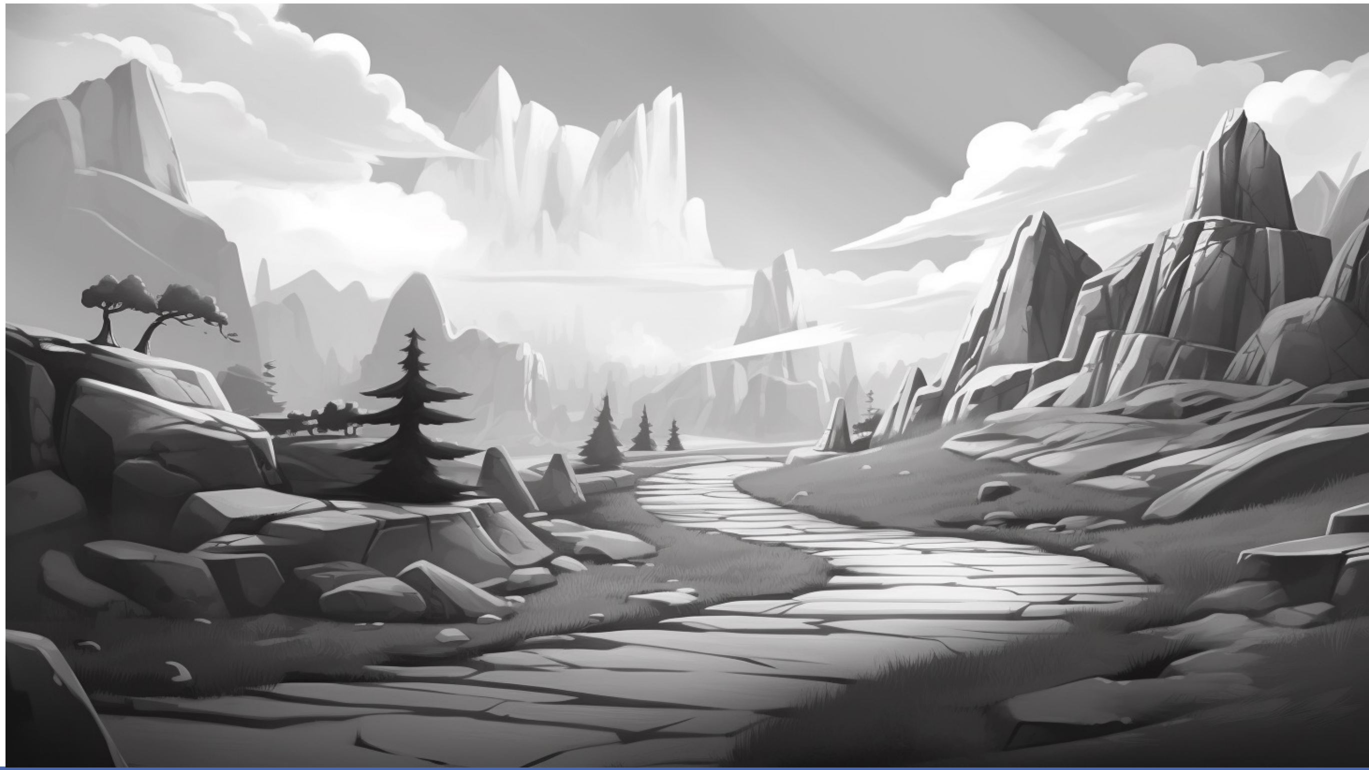
Paths & Practice
Learning and Development

- Self directed learning
- Learning by sharing
- Learn by doing
- Certification



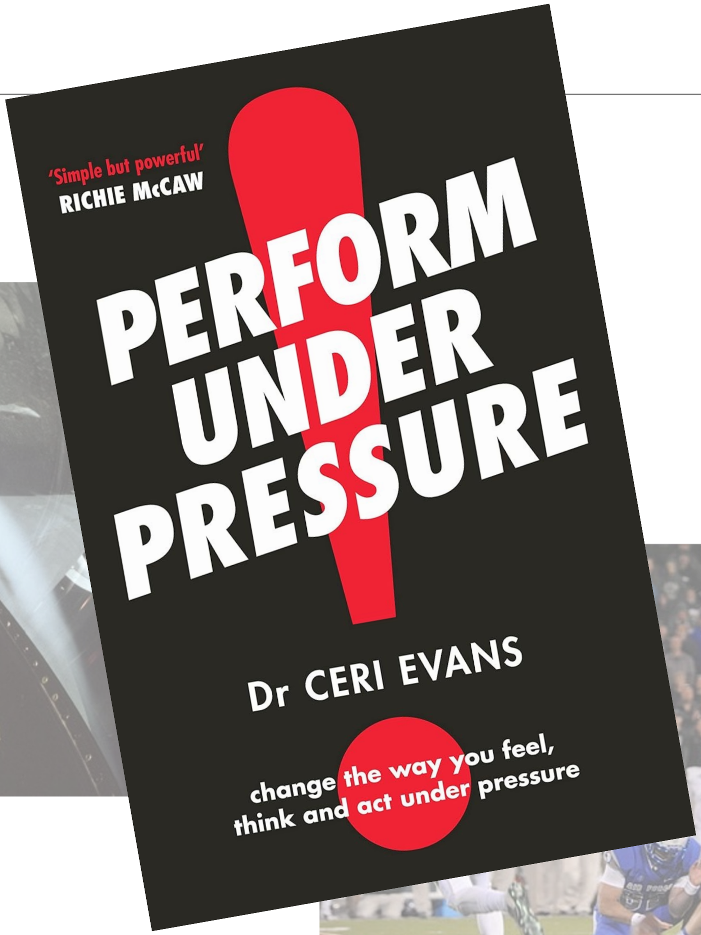
Paths & Practice Performance

- Same quality, but different speeds
- → overload the stars
- → can’t make plans
- Path to performance



Paths & Practice Performance

Paths & Practice Performance


'Simple but powerful'
RICHIE McCAW

PERFORM UNDER PRESSURE

Dr CERI EVANS

change the way you feel, think and act under pressure

GET
BETTER
SOON
I KNOW YOU'RE NOT SICK,
I JUST THINK YOU CAN BE BETTER.

Paths & Practice Performance

- Identify your “Performance Moments”
- Agree terms of engagement
- Practice!

Paths & Practice Performance

- Identify
  - Incident Response
- Agree
  - Triage, Escalate, Resolve
  - 5 targeted events
- Practice!
  - 5min rounds
  - Fake incident
  - Real tooling

Paths & Practice Performance

- Performance Moments
  - Incident Response
  - Project Planning
  - Risk Management
  - Crucial Conversations
  - Pairing

Paths & Practice

PRINCIPLE

- Competency (What)
- Performance (How)
- Alignment (Why)

IMPLEMENTATION

- Learning and Development
- Practiced Performance
- Driver Trees

~~Leadership Challenges~~

~~Start Up Challenges~~

~~Direction~~

~~Teamwork Challenges~~

~~Maps and Measures~~

~~Org Challenges~~

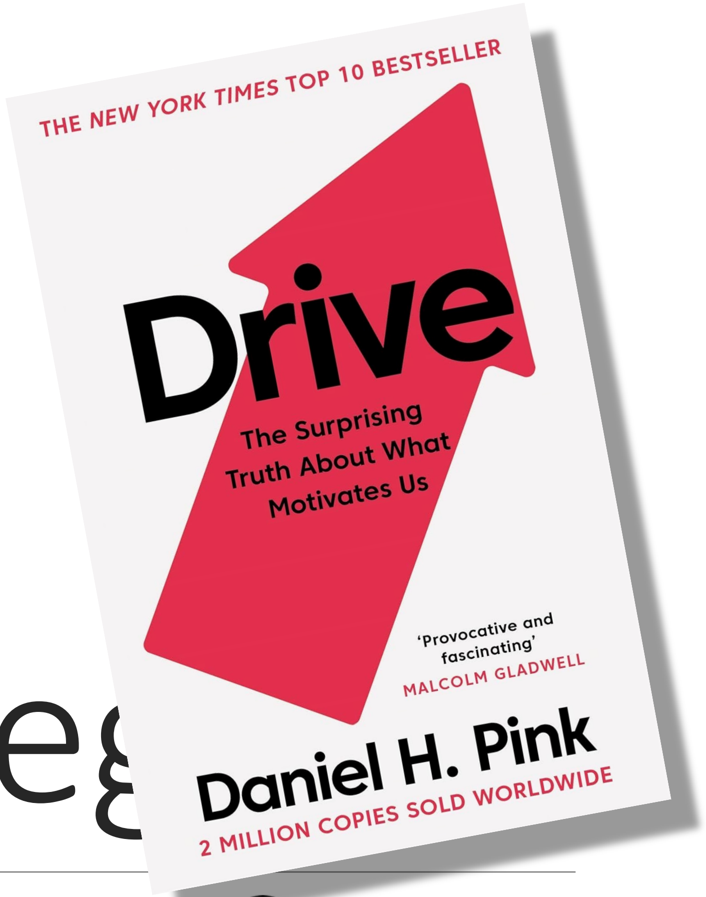
~~Driver Trees~~

~~Paths and Practice~~

Coordinated Strategy

Coordinated Strategy

Coordinated Strateg


THE NEW YORK TIMES TOP 10 BESTSELLER
Drive
The Surprising Truth About What Motivates Us
'Provocative and fascinating'
MALCOLM GLADWELL
Daniel H. Pink
2 MILLION COPIES SOLD WORLDWIDE

Team Success

PEOPLE’S ISSUES

- Being trusted
- Being valued
- Being aligned

LEADERS’ ISSUES

- ***Competency***
- ***Performance***
- ***Alignment***

Coordinated Strategy

Competence

- Reference Examples
- Competency Matrix
- Learning and Development

Performance

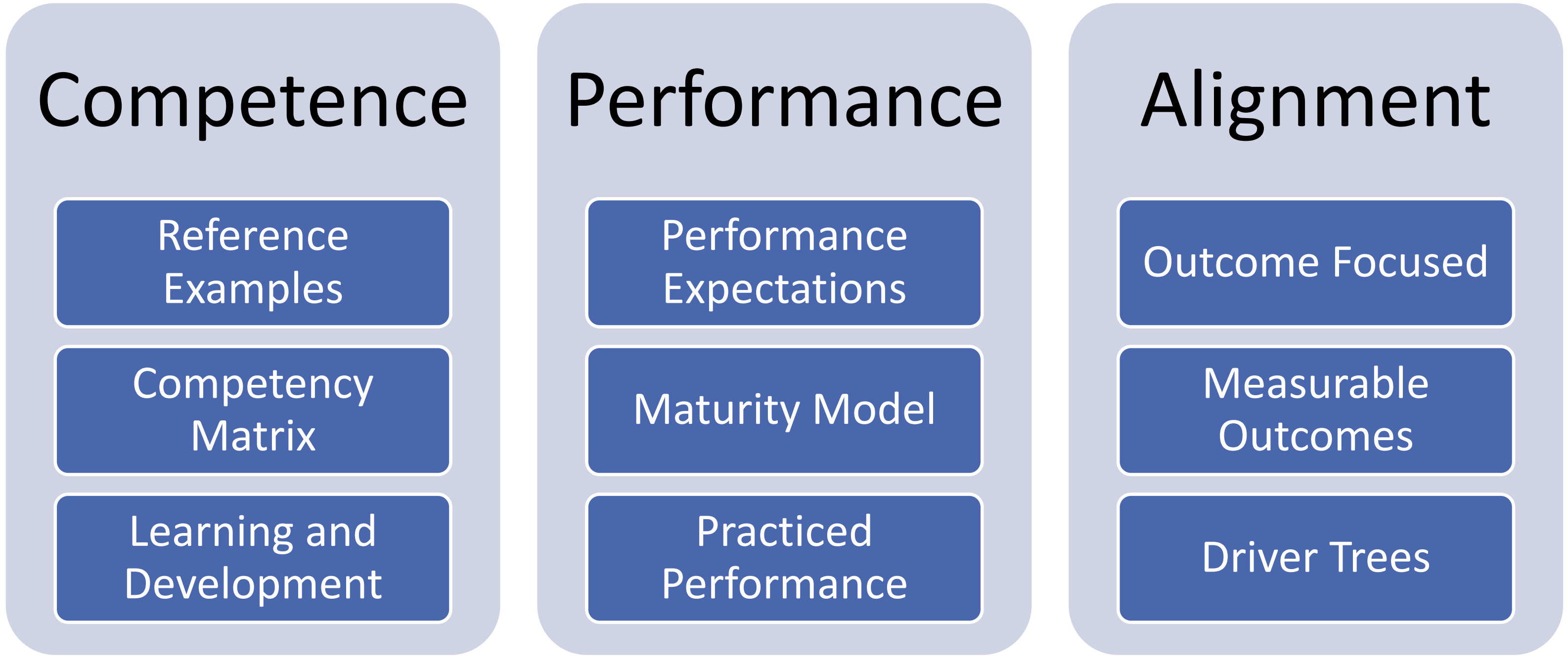
- Performance Expectations
- Maturity Model
- Practiced Performance

Alignment

- Outcome Focused
- Measurable Outcomes
- Driver Trees

LeeCampbell.com

Layers of Leadership



Questions….?

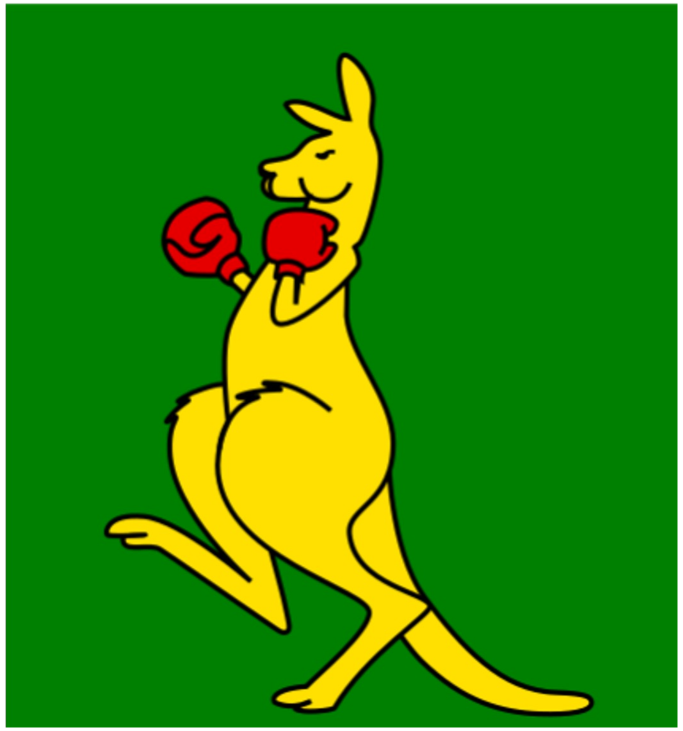
LeeCampbell.com

Lee's list

- ***Competency***
- ***Performance***
- ***Alignment***

Rhys' (Lee's Twin) Ozzie Translation

- ***Be fully sick***
- ***Crush it***
- ***Don't stuff up, mate***



Wardley Maps vs Driver Trees

DRIVER TREES

Understand the business better

Shared language

Focus on **measures**

WARDLEY MAPS

Understand the business better

Shared language

Focus on **needs**

Design From Genesis to Commodity