Vibe Coding

The New Era of AI-First Development

> "I’m fully giving in to the vibes and forgetting that the code even exists... I’m not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works." — **Andrej Karpathy**

Goals of this Presentation

1. **Broad Overview** - Understanding the landscape of players and paradigms
2. **Establish a Recommended Stack** - Aligning on the best tools for our team
3. **Daily Workflow Adoption** - Moving from "experimenting" to "integrated"
4. **Mandatory Quality Gates** - Standardizing AI usage in our PR process
5. **Security & Consistency** - Mitigating risks and bridging the skill gap

Current State of AI in the Team

The Consistency Gap

- **Power Users:** No of us fully explored what is possible. Landscape changes quickly
- **Experimentalists:** Use it occasionally for unit tests or boilerplate.
- **Newcomers:** Not using AI tools at all, missing out on massive productivity gains.
- **Result:** Inconsistent code quality, varying velocity, and knowledge silos.

The Shadow AI Risk

Security & Data Leakage

- **The Problem:** People are using unapproved tools/sites (ChatGPT, random extensions).
- **The Risk:** Sensitive company data, API keys, and trade secrets are leaking to external providers.
- **The Solution:** Standardizing on **GitHub Copilot** as our provider and **OpenCode** as our agent ensures:
  - Enterprise-grade data protection
  - Controlled context sharing

The Shadow AI Risk

> **Strict Policy:** The use of non-approved AI coding tools for company work is **strictly forbidden**. This is not just a preference; it is a critical security requirement to protect our intellectual property.

Agenda

> "I've never felt this much behind as a programmer." — **Andrej Karpathy**

1. **The AI Landscape** - Providers, Models, and Servers
2. **Paradigms** - How AI assists coding today
3. **Our Recommended Stack** - What we're using
4. **OpenCode Deep Dive** - Installation, Usage, Advanced Features
5. **GitHub Copilot Integration** - PR Reviews & Workflows
6. **What's Required vs Recommended**
7. **Q&A**

The AI Landscape

LLM Providers

Provider	Notable Models	Strengths
Anthropic	Claude Opus, Sonnet, Haiku	Reasoning, coding, safety
OpenAI	GPT-4o, GPT-5.2, o1/o3	General purpose, vision
Google	Gemini 3 Pro/Flash	Long context, multimodal
Meta	Llama 3.3, Llama 4	Open source, on-prem
Mistral	Mistral Large, Codestral	European, open weights
DeepSeek	DeepSeek-V3, R1	Cost-effective, reasoning

Model Features: The Trade-offs

Feature	High Reasoning (Opus/o1/R1)	Fast Execution (Flash/Haiku)
Accuracy	High (Complex Logic/Bugs)	Moderate (Boilerplate/Unit Tests)
Speed	Slow (Deep Thinking time)	Instant
Cost	Premium ($$$)	Commodity ($)
Context	Up to 200k tokens	**Up to 2M tokens**

Strategic Choices:

- **Thinking (System 2):** Use for architecture, refactoring, and hard-to-find bugs.
- **Context:** Use Gemini for full-repo analysis where Opus/o1 would truncate.
- **Price:** Flash is 10-50x cheaper; use it for simple "build" tasks.

Monitoring Your Usage

Track your premium model entitlement and feature access in real-time to balance reasoning vs. speed.

- **URL:** github.com/settings/copilot/features
- **Tip:** Monitor your "Premium requests" bar to manage your monthly quota effectively.

Model Servers (API Providers)

The same model can be served by different providers:

Server	What They Offer
OpenRouter	Unified API for 100+ models
Together AI	Fast inference, open models
Fireworks	Low-latency, fine-tuning
Groq	Ultra-fast inference (LPU)
Gemini (Google)	Massive context, multimodal
GitHub Copilot	Router to multiple models

> Tip: Check models.dev for model comparisons

AI Coding Paradigms

1. Completions (Autocomplete)

What: Inline suggestions as you type

Tools: GitHub Copilot (in VSCode), Codeium, Tabnine, Supermaven

Best for: Fast, low-friction suggestions

```python
def calculate_total(items):
    # AI suggests the rest as you type
    return sum(item.price * item.quantity for item in items)
```

2. Chat-Based Assistants

What: Conversational interface for coding questions

Tools: VSCode Copilot Chat, ChatGPT, Claude.ai, Gemini

Best for: Explaining code, brainstorming, learning

```
You: How do I optimize this SQL query?
AI: Here are 3 approaches...
```

3. Agentic Coding

What: AI that can read, write, and execute code autonomously

Tools: OpenCode, Claude Code, Cursor, Windsurf, Aider, Cline

Best for: Complex multi-file changes, refactoring, new features

```
You: Add authentication to the /settings route
AI: [reads codebase] [writes files] [runs tests] Done!
```

4. GitHub-Integrated Workflows

What: AI embedded in your Git workflow

Tools: Copilot Code Review, Copilot Coding Agent, OpenCode GitHub Action

Best for: PR reviews, issue triage, automated fixes

```
/opencode fix this issue
```

AI creates a branch, implements fix, opens PR

Paradigm Comparison

Paradigm	Autonomy	Context	Best Use Case
Completions	Low	Single file	Speed
Chat	Medium	You provide	Learning
Agentic	High	Full codebase	Complex tasks
GitHub	High	PR/Issue	Workflow automation

Use ALL of them strategically

Our Recommended Stack

The Setup

```
                    +------------------+
                    |    OpenCode      |  <-- Primary coding agent
                    +--------+---------+
                             |
                    +--------v---------+
                    | GitHub Copilot   |  <-- Router/Provider
                    +--------+---------+
                             |
          +------------------+------------------+
          |                                     |
+---------v---------+               +-----------v---------+
|  Claude Opus 4    |               |  Gemini 3 Flash     |
|  (Complex tasks)  |               |  (Fast operations)  |
+-------------------+               +---------------------+
```

Why This Stack?

- **OpenCode** - Open source, **IDE-agnostic**, terminal-native (works everywhere, including VSCode terminal and SSH)
- **GitHub Copilot as Router** - Single billing, access to multiple models
- **Claude Opus** - Best-in-class for complex reasoning and coding
- **Gemini 3 Flash** - Massive context window (2M tokens) and **9x cheaper than Opus**, ideal for codebase exploration

Plus: GitHub Copilot for PR reviews and inline completions

OpenCode Deep Dive

What is OpenCode?

- Open source AI coding agent
- Terminal-based (TUI), Desktop app, or **VSCode extension**
- Works with any LLM provider
- Full codebase awareness
- Undo/Redo changes
- Share conversations with team

Installation

```
# Install script (easiest)
curl -fsSL https://opencode.ai/install | bash

# Or via npm
npm install -g opencode-ai

# Or via Homebrew (macOS/Linux)
brew install anomalyco/tap/opencode

# Verify installation
opencode --version
```

First Run

```
# Navigate to your project
cd /path/to/project

# Launch OpenCode
opencode

# Initialize for this project
/init

# Connect to a provider
/connect
```

Note: `/init` creates `AGENTS.md` - this is where we codify our team's coding standards and project-specific rules. **Commit this file!**

AGENTS.md - Your AI's Instructions

The `/init` command creates an `AGENTS.md` file in your project root.

What it does:

- Provides project-specific context to the AI agent
- Codifies your team's coding standards and conventions
- Defines domain knowledge the AI should apply

AGENTS.md Example

Example:

```
# Project: Photonic Circuit Simulator

## Domain Context
- We simulate silicon photonic integrated circuits (PICs)
- Units: wavelength in nm, distances in μm, power in dBm

## Code Standards
- Use gdsfactory for layout generation, Meep for FDTD simulations
- Use NumPy for matrix operations, avoid loops over ports
- All physical constants must come from scipy.constants
```

> **Commit `AGENTS.md`** - It becomes shared domain knowledge for the whole team's AI workflows.

Example: https://github.com/wave-photonics/foundry/blob/update-docs/AGENTS.md

🚀 Live Demo

See it in Action

Basic Usage: The Two Modes

Always Plan Before You Build

Mode	Key	Purpose	Reasoning Model
Plan	`Tab`	Read-only, suggests approach	**Claude Opus** (High Reasoning)
Build	Default	Full access, makes changes	**Gemini 3 Flash** (Speed/Context)

> **Standard Workflow:**
>
> 1. Start in **Plan** mode to verify context and strategy.
> 2. **Chat & Iterate** with the agent until the plan is solid.
> 3. Switch to **Build** to execute the approved plan.

Asking Questions

Use `@` to reference files:

```
How is authentication handled in @src/api/auth.ts?
```

Drag & drop images for visual context:

```
Use this design mockup as reference [drag image]
```

Making Changes

```
Add rate limiting to the /api/users endpoint.
Look at how it's done in @src/api/products.ts
```

OpenCode will:

1. Read the codebase
2. Show you a plan
3. Make the changes
4. You review and approve

Undo/Redo

Made a mistake? No problem.

```
/undo    # Revert last changes
/redo    # Restore reverted changes
```

Can be run multiple times to step through history.

The Two Default Agents - Planning

1. **Plan Agent (The Thinker)**
    - Read-only mode
    - Analyzes and suggests
    - **Chat & Iterate** until the approach is verified
    - Powered by **Claude Opus** for complex reasoning

The Two Default Agents - Building

2. Build Agent (The Doer)

- All tools enabled
- File read/write, bash commands
- Powered by **Gemini 3 Flash** for fast execution
- Used only after you've approved a plan

Switch with `Tab` key

Subagents

OpenCode spawns specialized agents for tasks:

General - Research, multi-step tasks
Explore - Fast codebase navigation

Invoke manually with `@`:

```
@explore find all API endpoints in this project
```

Advanced Features

Custom Agents

Create specialized agents in `.opencode/agent/`:

```
# .opencode/agent/security-auditor.md
---
description: Performs security audits
mode: subagent
tools:
  write: false
  edit: false
---

Focus on: input validation, auth flaws,
data exposure, dependency vulnerabilities.
```

MCP Servers (Model Context Protocol)

Extend OpenCode with external tools:

```
// opencode.json
{
  "mcp": {
    "github": {
      "type": "remote",
      "url": "https://api.githubcopilot.com/mcp/"
    },
    "context": {
      "type": "remote",
      "url": "https://mcp.context7.com/mcp"
    }
  }
}
```

Then: `Why did the CI fail on my last commit? use github`

Skills

Reusable instruction sets in `.opencode/skill/`:

```
# .opencode/skill/git-release/SKILL.md
---
name: git-release
description: Create consistent releases
---

## What I do
- Draft release notes from merged PRs
- Propose a version bump
- Provide gh release create command
```

Agent loads skill when needed via `skill({ name: "git-release" })`

Permissions

Control what OpenCode can do:

```
{
  "permission": {
    "bash": {
      "*": "ask",
      "git status": "allow",
      "rm *": "deny"
    },
    "edit": "allow"
  }
}
```

allow - Run without approval

ask - Prompt for approval

deny - Block the action

GitHub Integration

OpenCode in GitHub

Add OpenCode to your GitHub workflow:

```
opencode github install
```

Then in any issue or PR comment:

```
/opencode explain this issue
```

```
/opencode fix this
```

OpenCode runs in GitHub Actions, creates branches, opens PRs.

GitHub Copilot Features

Copilot Chat (VSCode) - Ask questions about your code directly in the IDE

Completions - Inline suggestions in VSCode

PR Descriptions - Auto-generate PR summaries

Code Review - AI-powered review suggestions

Coding Agent - Autonomous PR creation (Pro+/Business)

Copilot in PRs (MANDATORY)

When reviewing PRs, use Copilot to:

1. **Generate summaries** - Click "Copilot" button on PR
2. **Request AI review** - Add Copilot as a reviewer
3. **Ask questions** - Use Copilot Chat in PR context

```
@copilot explain what this change does
@copilot are there any security concerns?
```

Required vs Recommended

Required: Quality Gate Mandate

Tool	Usage
GitHub Copilot	Required for all PR descriptions
Copilot Code Review	**Mandatory** reviewer on all PRs
OpenCode	Primary tool for complex coding tasks

> **Frame:** These aren't just tools; they are our **Automated Quality Gates**. Just like Linting and Unit Tests, AI Review is now part of our Definition of Done.

Recommended (Explore Further)

Tool	Why
Custom Agents	Standardize team workflows
MCP Servers	Connect to Sentry, Jira, etc.
Skills	Share reusable prompts
Copilot Coding Agent	Automated issue-to-PR workflow

Getting Started Checklist

- [ ] Install OpenCode: `curl -fsSL https://opencode.ai/install | bash`
- [ ] Run `/connect` and configure GitHub Copilot as provider
- [ ] Run `/init` in your project
- [ ] Try Plan mode (`Tab`) before Build mode
- [ ] Enable Copilot in VSCode
- [ ] Add Copilot as reviewer on your next PR

Future Directions

OpenCode Ecosystem - Growing community of plugins, themes, agents

Oh-My-OpenCode - Community configurations (coming soon)

Claude Code - Anthropic's official agent (enterprise pricing)

Your Contribution - Help shape company AI tooling recommendations!

Q&A

Questions?

Resources

- **OpenCode Docs**: https://opencode.ai/docs
- **GitHub Copilot Docs**: https://docs.github.com/copilot
- **Model Comparisons**: https://models.dev

Thank You!

Start using OpenCode today:

```
curl -fsSL https://opencode.ai/install | bash
```

Let's make AI a daily part of our coding workflow.