

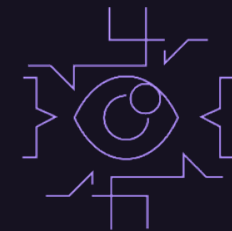
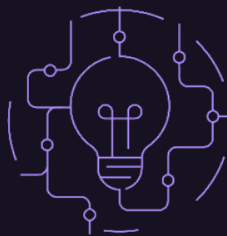


Accelerating Application Refactoring:

AI in DevSecOps

Joel Krooswyk
Federal CTO

Secure Software by Design
August 7, 2024



State of the Union in DevSecOps



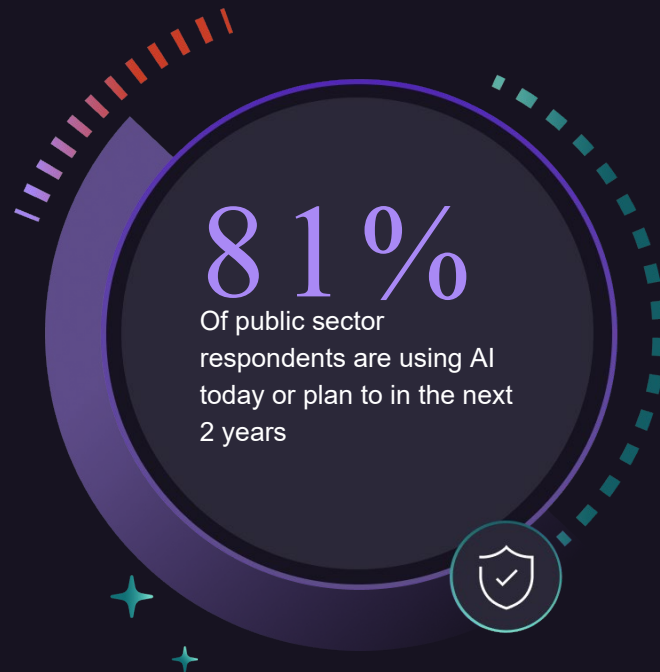
AI is already accelerating the SDLC

>50%

Reported efficiency gains in within months of AI implementation

5-10x

Targets for SDLC efficiency gains



Source: 2023 DevSecOps survey



With AI acceleration, can we reconsider legacy code?

Cloud Migration

Modernization

Performance

Scalability

Memory Safety

Compliance

Technical Debt

Unsecured Code

Vintage Vulnerabilities

Emergent Threats

Developer Knowledge

Cost Reduction



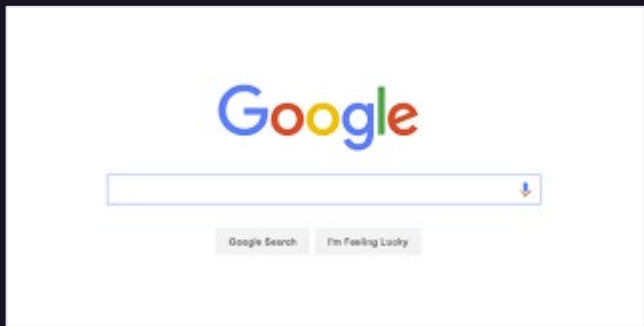
Defining refactoring

“Refactoring is a controlled technique for improving the design of an existing code base, a disciplined technique for restructuring an existing body of code, altering its internal structure without changing its external behavior.”

- Martin Fowler



Refactoring can drive incredible outcomes



Python -> C++
Now Rust?



Many -> Java






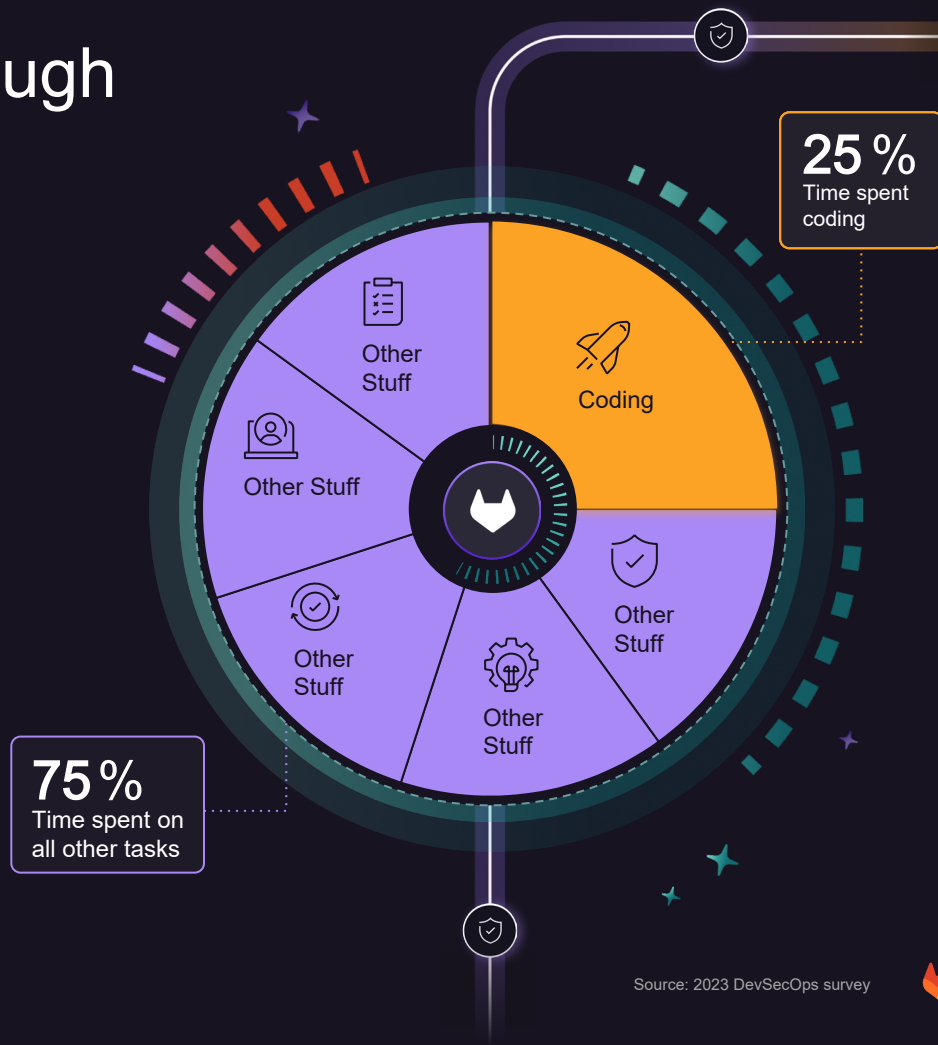
Perl -> Scala



Prioritizing refactoring is tough

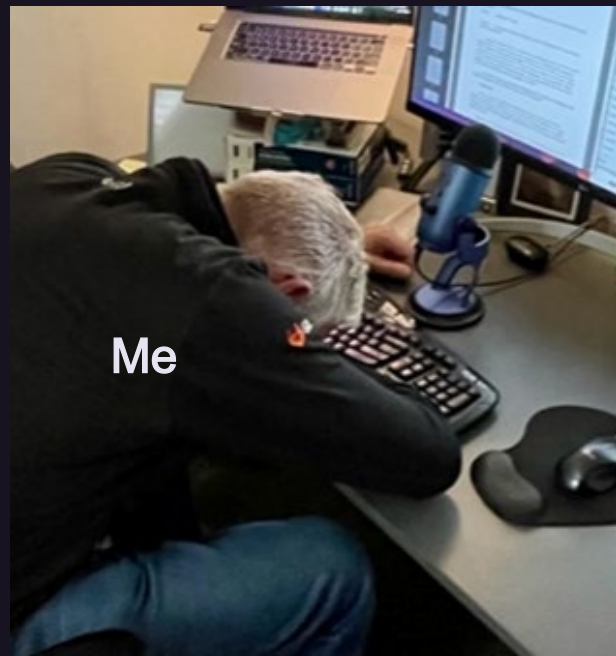
Refactoring requires

-  Sufficient developer time
-  Deep developer skills
-  Accurate testing



How do we refactor?

“The simplest way is to just clone the code and start hacking away improving the design.”



Common refactoring techniques

Red-Green

Refactor code until it passes the tests created at the start

Extract Method

Address complexity, clarity, and structure

Simplifying Method

Streamline method calls and expressions

Composing Method

Large code function fragmentation and breakdown

Abstraction

Scaled refactoring of huge projects

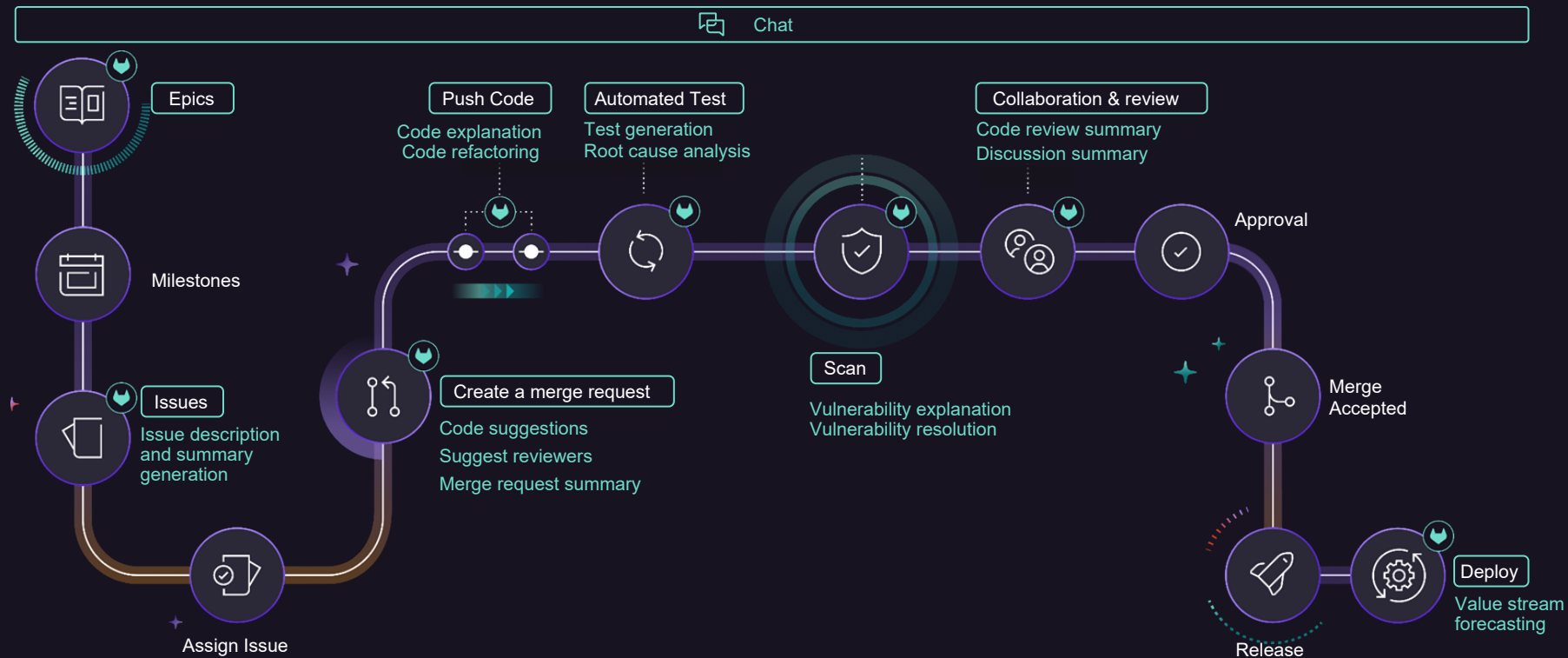
%&@#!



AI acceleration in a vacuum - code assist only

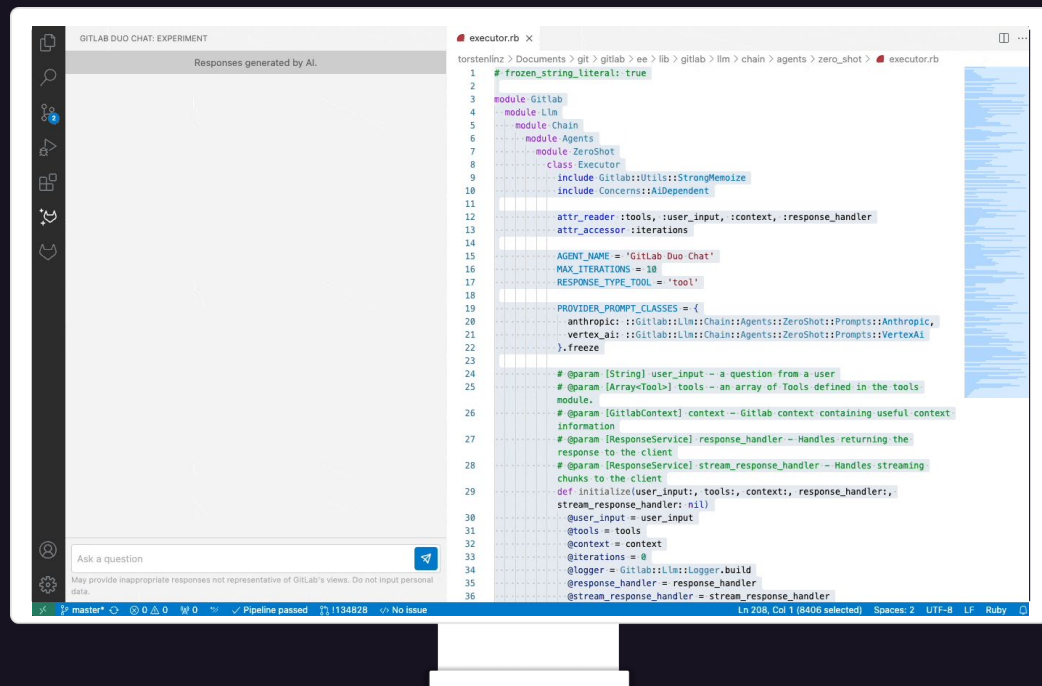


Applying AI throughout the SDLC



Refactoring with AI today

- 🕒 Explain existing code
- 🕒 Create new code
- 🕒 Create unit tests
- 🕒 Analyze root cause
- 🕒 Suggest fixes for failed tests
- 🕒 Remediate vulnerabilities



Code explanation helps drive AI prompt accuracy

Explain uncommented code functionality

Explain functionality in context of a familiar language

Explain coding patterns, syntax, calls, or methods

Explain this concept to me like I'm 5

Explain if I can do this work with my current environment
setup

Explain expected inputs and formats

Explain best practices for my new code

Tell me more about that



+ ≡ DevRel - explain code

Model settings



Model

claude-3-opus-2024 v

Temperature

0

Max tokens to sample

1000

System Prompt



You are a programming language expert.

When the user asks a question, they will add the source code snippet at the end, starting with "Source:". Analyze the source code, and focus on explaining algorithms and functionality.

If the user mentions "security", focus on potential vulnerabilities instead.

USER

```
Explain the following source: // Create a client for
Kubernetes observability
// Create a function that
// Reads the kubernetes configuration file from the
KUBECONFIG env var
// Create kubernetes context, namespace default
// Inspect container, pod, service status and print an
overview
// Import necessary packages
// Create main package
package main
```





RESPONSE

This Go source code creates a client f
functionality:

1. The code imports necessary pack
metav1 package for Kubernetes A
2. The `main` function is the entry p
to create a Kubernetes clientset, :
functions to print information abo
3. The `getClientset` function rea
`KUBECONFIG` environment variab
`clientcmd.BuildConfigFromF`
`kubernetes.NewForConfig`. If
4. The `printPods` function takes t
clientset to list all pods in the spe
`clientset.CoreV1().Pods(na`
namespace.
5. The `printServices` function is
namespace using `clientset.Cc`



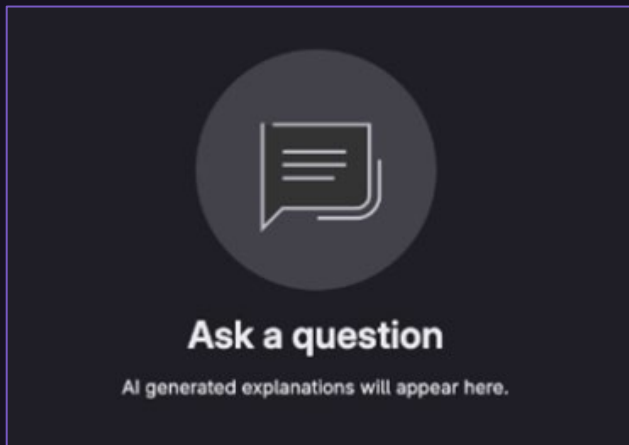
Prompt effectively for code assistance

-  Be conversational
-  Provide as much context as possible
-  Ask for different approaches to solve a problem
-  Prompt with a pattern

State the problem

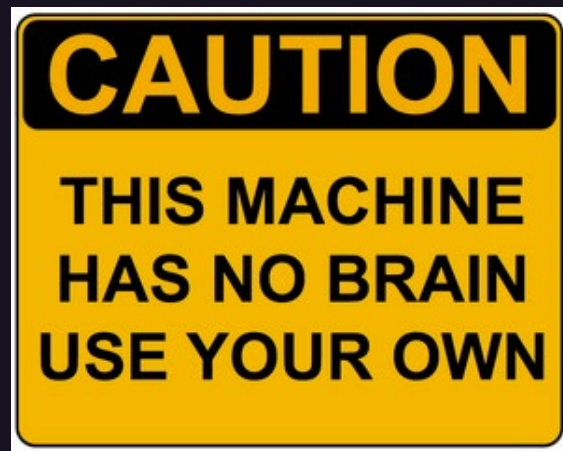
Ask for help

Provide additional requests



AI responses require caution

- ✓ Non-determinism: repeated prompts=different outcomes
- ✓ Incorrect results: context is critical
- ✓ Security problems: vulnerability origination
- ✓ Big bang failure: going beyond small iterations
- ✓ Logic mistakes: AI does better with fuzzy I/O
- ✓ Human checkpoints and gates are required



AI can test its own code

Examples



Test case creation



Failed test remediation



Pipeline root cause analysis

AI supports TDD



AI requires full context up front



AI requires human validation

ANTHROPIC

Dashboard Workbench

DevRel - explain code

Model settings

Model

claude-3-opus-2024

Temperature

0

Max tokens to sample

1000

System Prompt

You are a programming language expert.

When the user starts the question with "Generate tests", analyze the source code after the keyword "source:" and use modern development test frameworks to generate tests. Focus on extreme cases, and regression testing.

USER

```
Generate tests for the following source: // Create a client
for Kubernetes observability
// Create a function that
// Reads the kubernetes configuration file from the
KUBECONFIG env var
// Create kubernetes context, namespace default
// Inspect container, pod, service status and print an
overview
// Import necessary packages
// Create main package
package main
import (
    "context"
    "fmt"
    "os"
    "k8s.io/client-go/kubernetes"
    "k8s.io/client-go/tools/clientcmd"
    metav1 "k8s.io/apimachinery/pkg/apis/meta/v1"
)
func main() {
    clientset := getClientset()
    namespace := "default"
    printPods(clientset, namespace)
    printServices(clientset, namespace)
    printContainers(clientset, namespace)
}
```

Add Messages

View all API options

3 of 3



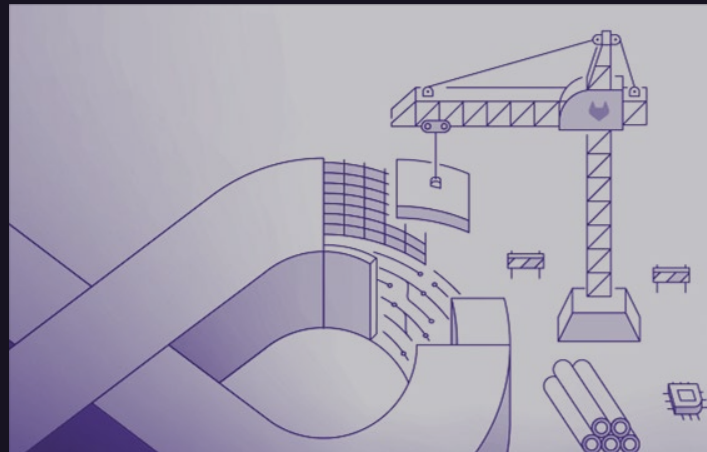
AI-assisted code security reduces shift - left fatigue

AI security capabilities can:

- 🕒 Explain vulnerabilities
- 🕒 Automatically remediate vulnerabilities
- 🕒 Remediate vulnerabilities across code bases

AI iterates on its own work while humans:

- 👤 Manage the process
- 👤 Audit the work
- 👤 Validate the output



Story: Legacy monolith to microservices in the cloud

What

Cloud Migration Initiative

Language

C to Rust

Architecture

Monolith to Microservices

```
func main() {  
    clientset := getClientset()  
    namespace := "default"  
  
    printPods(clientset, namespace)  
    printServices(clientset, namespace)  
    printContainers(clientset, namespace)  
}  
  
func getClientset() *kubernetes.Clientset {  
    kubeconfig := os.Getenv("KUBECONFIG")  
  
    config, err := clientcmd.BuildConfigFromFlags("", kubeconfig)  
    if err != nil {  
        panic(err)  
    }  
}
```

Outcomes



90% less code



50x more scalable



100x faster at scale



Scenario: moving off mainframes

What

Mainframe Modernization

Languages

- ⚙ Assembly to Java
- ⚙ SAS to Python
- ⚙ Cobol to Java or C#

Architecture

Mainframe to Cloud



Outcomes

- ⚙ Less code
- ⚙ Secured code base
- ⚙ Budget savings
- ⚙ Maintainability
- ⚙ Performance
- ⚙ Native API and AI use



Scenario: migrating embedded and firmware

What

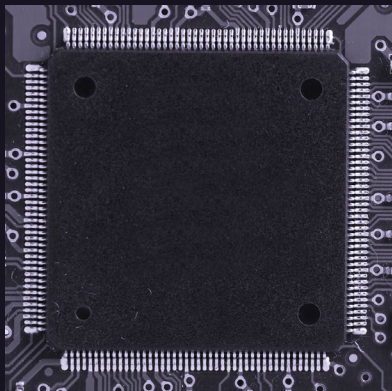
Firmware modernization

Languages

- ⚙️ ADA
- ⚙️ Embedded C

Architecture

Embedded processors



Outcomes


- ⚙️ Less code
- ⚙️ Memory Safety
- ⚙️ Maintainability
- ⚙️ Simplified build chain



The future of software development

GitLab Duo Workflow

 Fully automated workflows

 Autonomous agent







 Proactive AI

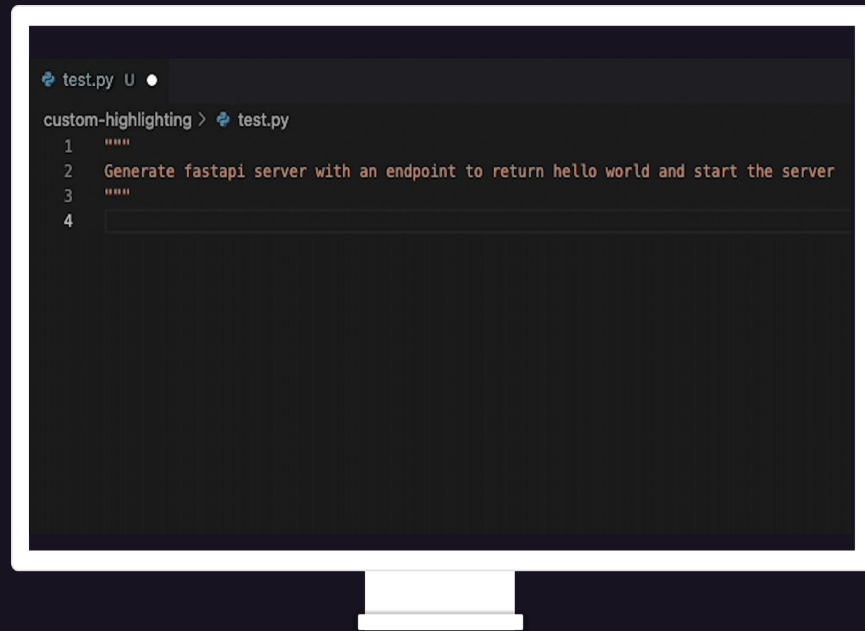
 Unified context

 Increased efficiency



Autonomous AI project workflow

-  Auto-configure projects
-  Create, plan, and prioritize tasks
-  Recommend work to teams/people
-  Create initial code drafts
-  Create and run tests
-  Remediate vulnerabilities



Thank you

