tmux enables AIs to operate servers safely

Posted 2026-03-17 | stdout

We've all seen plenty of horror stories about AI trashing servers. Yet, there are still tedious tasks we'd love for AI to handle. To keep things safe, you have to manually copy-paste back and forth commands and outputs. Yet the current mainstream solutions usually involve "adding another layer": relay IO, intercepting dangerous commands or even using a smaller model as a filter.

But these solutions rely heavily on dedicated Agent tools or MCP, which means you have to let the AI connect to server directly as first party. If the server doesn't allow direct SSH, sits behind a jump box, or is completely air-gapped, you're basically stuck.

My friend and I were discussing this, and at one point, we even thought about vibe coding a middleware to handle it. Then, while staring blankly at iTerm2 and Ghostty, it hit me: **tmux**.

If I connect to the server via tmux on my local machine, the rest is easy. Here's the prompt I used:

> I've started a new tmux session: `tmux new-session -d -s opus .`
> I've already logged into the server. This server has no external internet access but has mirrors for yum, pip, etc.
> The environment XXX and working directory XXX are ready. The objective is XXX
> First, analyze the environment and write a plan. If you have questions, clarify them first.

To my surprise, the AI actually started interacting and executing commands:

- **Sending commands:** `tmux send-keys -t opus 'complete_command' Enter`
- **Reading output:** `tmux capture-pane -t opus -p -S -15`

However, I soon realized `send-keys` isn't entirely safe—the AI would just append an `Enter` and execute the command immediately. What to do?

When in doubt, ask the AI. The conclusion: Replace `tmux send-keys` with a filtered alias.

```
ts() { tmux send-keys -t opus -l -- "$(printf '%s' "$*" | tr -d '\000-\037\177')"; }
```

So, the workflow looks like this:

1. Enable this alias, fire up tmux and login to the server
2. In your Agent config, allow `ts` to execute automatically, but disallow `tmux` .
3. Write your prompt, explaining what needs to be done and instructing the AI to use this alias.
4. The AI starts thinking, the commands would appear on your tmux. **Crucially, it absolutely cannot press Enter.**
5. Human-in-the-loop: Stare at the command carefully ⚠️. If the command looks safe, you press Enter to proceed.
6. If there's a problem, hit `Ctrl+C` , and start a new line with a comment: `# I canceled this because blah` .
7. Go back to the Agent. Since `tmux` is blacklisted for auto-execution, you manually "Allow" it to run `tmux capture-pane` to read the output.
8. Iterate until the task is complete.

You don't need to bookmark the `ts` alias, you can always ask your AI to make one for you. The alias isn't perfect in all cases, but clever AIs would figure it out 😉

Heck I don't even bother to explain the `ts` here. But I did asked several AIs to check for correctness and rubustness.

P.S. if you're on a "# of Requests" subscription plan, this entire operation theoretically only counts as **one**.
P.P.S. here tmux acts as a natural "checkpoint" for autoregressive generation process. You can approve, cancel or redirect

---

□AI撸服务器翻车的案例很多了。但是服务器上有些麻烦事儿还是想让 AI去解决，为了安全，网上现在的方案都是——再套一层。比如拦截危险指令，□小模型做过滤等等。

这些方案都依靠agent tool或者MCP，也就是说你得允许让AI直连。如果服务器不允许直接ssh，有跳板机，断外网的话可能就抓瞎了

我和朋友也聊到这个问题，甚至一度想 vibe 一个middleware去实现这样的功能。我盯着 iTerm2 和 Ghostty 发呆，突然想到个东西，tmux。

在本机□tmux把服务器连上，然后接下来的问题就简单了，下面是我的 prompt

> 我新建了个tmux
>
> tmux new-session -d -s opus
>
> 并且已经登录服务器。该服务器禁止外网，但是有 yum pip 等镜像。
>
> 已经准备好XXX，工作目录 XXX，需要实现 已经准备好XXX，工作目录
>
> 先分析环境，写个plan。有问题确认清楚

没想到 AI 真的开始调□命令开始交互执行了

- 发送指令： `tmux send-keys -t opus '□整命令' Enter`
- 阅读输出： `tmux capture-pane -t opus -p -S -15`

期间发现 `send-keys` 也不太安全，AI直接把命令回车执行了。怎么办？

遇到这种问题，继续问AI啊。结论是 `tmux send-keys` 换成一个带过滤 alias 就行

```
ts() { tmux send-keys -t opus -l -- "$(printf '%s' "$*" | tr -d '\000-\037\177')"; }
```

所以流程是：

1. 把tmux跑起来，把这个 alias 生效
2. 在 agent 配置里，允许 `ts` 直接执行，不允许 `tmux` 改成手动确认
3. 写prompt交代你要干啥，交代务必□这个alias。
4. AI开始想办法，一顿命令输出，在tmux敲命令了，但是重点来了，绝对没按回车
5. □瞪眼法观察里面的是否有诈，human-in-the-loop。没问题就敲回车。
6. 有问题你Ctrl+C然后新起一行写个注释 `# I canceled this because blah`
7. 回到 agent 工具，这个时候 `tmux` 命令应该是黑名单，不能直接执行，你点 allow 允许它去 `tmux capture-pane` 读输出内容
8. 逐渐迭代直到任务完成

这套流程还有一个额外的好处，如果你买的是「次数」套餐，那么整体操作下来，理论上只算1次。

Comments

Go