We have been using Azure DevOps self-hosted build agents until now, which have the advantage of being tightly integrated with Azure, to the extent of being almost entirely hands off
– ADO automatically orchestrates the creation and deletion of build agents before and after
pipeline runs. Would it be possible to replicate orchestration of self-hosted runners with GitHub
as it is in ADO?

Questions

- Do you have Infrastructure as Code skillsets between your current team?

Yes, GitHub Actions does support self-hosted runners, and you can replicate much of the orchestration found in Azure DevOps (ADO). However, the level of automation is not exactly the same out of the box. In GitHub Actions, you need to manually manage the lifecycle of your self-hosted runners, such as their creation and deletion after pipeline runs.
You can automate this using Infrastructure as Code (IaC) tools, like OpenTofu, or cloud-specific solutions, such as auto-scaling groups in AWS, GCP, or Azure itself, which I highly recommend in your case.

To mirror the automated orchestration of build agents, you could set up self-hosted runners in an auto-scaling group in your cloud provider, allowing instances to be dynamically spun up or down as needed. This way, you could integrate the process more tightly with GitHub Actions, achieving a similar hands-off approach as in ADO.

Also, you could use ARC (Actions Runner Controller) which is a Kubernetes-based piece of software, leveraging the operator pattern of k8s, allowing you to seamlessly manage scaling runners up and down.

Rationale

GitHub's self-hosted runners provide great flexibility and control, but they can lack the native automation for scaling up and down runners post-build like Azure DevOps. Instead of discouraging my customer to reach their goal, I want them to achieve the same experience as ADO so I found a solution that involves the use of open source tools, including one maintained by the GitHub actions team in collaboration with other independent maintainers.
In this way I provide a viable alternative that can keep the customer happy with our product and can enhance their trust in our regards as we are not pushing them to move to the Cloud but instead we are respecting their on-prem requirements.

Sources:
- https://github.com/actions/actions-runner-controller

-
https://docs.github.com/en/actions/hosting-your-own-runners/managing-self-hosted-runners/about-self-hosted-runners
-
https://docs.github.com/en/actions/using-github-hosted-runners/using-larger-runners/about-larger-runners (just checked about the larger runner autoscaling but it could not be feasible for this specific customer)
- My experience with OpenTofu, IaC and automation of scaling groups for self-hosted platform solutions (For example, I managed Atlassian Jira, JFrog's Artifactory and Puppet, on prem)

We are considering both your hosted (GitHub Enterprise Cloud) and on-premises (GitHub Enterprise Server) versions of GitHub Enterprise for our team of 60 developers. Which one is the better choice for me to present to management?

Questions
- What do you feel is the most important feature for your management?

For a team of 60 developers, GitHub Enterprise Cloud (hosted) is often the better option due to its lower operational overhead and the built-in scalability. GitHub Enterprise Cloud, allows you to not worry about server maintenance, backups, updates, or monitor your infrastructure costs. Every aspect of the platform integrates natively with other GitHub services like GitHub Actions, Codespaces, and Copilot.

On the other hand, GitHub Enterprise Server (on-premises) may be preferable if your organization has any strict regulatory or compliance requirements (example: government agencies or industrial patent data) that require information to be stored on-premises, or in cases where you need custom integrations that cannot be achieved in the cloud. For example if your workflow involves edge-networks or air-gapped environments.

Rationale
In my experience as a Cloud Engineer, for most teams, cloud-hosted services are more often than not cost-effective and allow for no operational overhead, even in cases of extreme scaling.

On-premises solutions could and should be chosen for specific business cases where compliance, or security requirements are stricter. This is true regardless of the operational overhead they bring in, since the risk of having privacy-sensitive data in public cloud environments can be too high, especially if the customers are operating in domains such as Military, Defense, Governmental bodies/agencies.

Sources
- https://docs.github.com/en/get-started/onboarding/getting-started-with-github-enterprise-cloud

- My experience as a GitHub Enterprise Cloud user (org management level), that gave me a good overview of its features and trade-offs.

We have provided 200 Copilot seats for developers across the whole organization, but according to the weekly unique user engagement metrics, I see that only 50 of the developers
have actually used Copilot in the last 3 weeks. Can you help us cancel the 150 licenses that are
not being used? Can we do that programmatically?

Questions
- Why do you think those seats are left unused?
- How long ago did you start using Copilot?
- What does the onboarding look like for a new developer in your company / how do you currently onboard people on Copilot?

Before proceeding with cancellation, I think it would be beneficial to first assess and explore why those developers are not using the remaining 150 Copilot licenses.
We could investigate and check if they received proper training and, if not, schedule demo sessions with them to showcase the features of Copilot and how it would help them to get up to speed with their development flow.
We should also check if they have access to the necessary integrations they want and to achieve that, we could survey what integrations they would be looking for.

Anyway the answer is definitely yes, it is possible to cancel unused Copilot seats programmatically: GitHub provides an API that allows administrators to manage both licenses and seats.
You can also monitor the current seat usage through another metrics API and the GitHub dashboard UI to then automate the cancellation of unused seats using a script that interacts with the licensing APIs.

Rationale
I discovered that programmatic management of licenses is possible via GitHub's API for Copilot, by looking at the docs. However, I would say it's always a good idea to first explore with the customer why the usage could be low in order to avoid prematurely canceling licenses that may be proven useful to them in the future, to only then reactivate them further down the line.
On top of keeping the account value at a certain level which would benefit the organization too.
That said, I would never think of locking a customer in by denying access to information such as “how to delete a license”, I firmly believe that transparency and integrity is important to maintain a good supplier-customer relationship and that is an investment in the long run, much more valuable than any short-term squeeze.

Sources
-
https://docs.github.com/en/copilot/managing-copilot/managing-github-copilot-in-your-organization/managing-access-to-github-copilot-in-your-organization/revoking-access-to-copilot-for-members-of-your-organization
-
https://docs.github.com/en/enterprise-cloud@latest/rest/enterprise-admin/license?apiVersion=2022-11-28#list-enterprise-consumed-licenses

Is there a way to render the content on our Enterprise Managed Users GitHub instance with the logo of our company? We would also like to change the color scheme so it reflects the
identity of our company. Could we do that by injecting a custom JavaScript snippet?

It's surely possible to have your company logo by uploading it on the Enterprise Settings panel but for any other kind of customization, after researching and consulting with the product team, it is not possible to change the theme or the color palette (aside from light/dark mode).

Reflecting on your proposal to inject a custom js snippet that would unfortunately be a security-sensitive operation that is deemed to be not feasible by the product team.
To better reflect your company identity, what I might suggest is to use your band assets and logo in the readme.md files of both your organization (in a .github repository) and each of your repositories.
In this way you can achieve branding without compromising on the security of the platform!

Rationale
After researching on the docs if there's any way to change palette and theme of the GitHub Enterprise UI, I concluded that it is not possible, so I explored a different solution instead, that could suite the customer need without compromising on security - as the injection of custom js snippets could lead to execution of insecure code on the client's side -
Another thing that I would definitely do is communicating with the product team and reporting back the customer's feedback so that they can prioritize customization options on the product roadmap as these would benefit every other customer too.

- Sources:
https://docs.github.com/en/enterprise-cloud@latest/organizations/collaborating-with-groups-in-organizations/customizing-your-organizations-profile
https://docs.github.com/en/get-started/accessibility/managing-your-theme-settings