Autonomous

1. Abstract

Autonomous; a platform which provides decentralised autonomous deployment for software and hardware using cloud infrastructure. By decentralising the control of deployments, changes to services can be made at any time and its infrastructure can be funded, owned and controlled by decentralised communities.

Additionally, this platform is capable of itself being controlled by a decentralised colony, whilst still providing services to control the deployments in other colonies. In short, the platform which controls colony infrastructure is itself controlled by a colony, and can update itself: **Autonomous can manage and run itself through a shared, open source colony and maintain decentralised control of the platform.**

2. Introduction

Nearly all modern organisations own infrastructure in the form of software or hardware; this infrastructure may be powering services, storing vital business data or performing other vital functions. Typically, these organisations operate in high-trust environments and require physical access to these computational resources by owning machines or credentials to cloud environments.

With the rise of blockchain technologies, open source communities and the emerging popularity of decentralised organisations, there is a significant need for adapting how infrastructure can be owned by groups, especially in limited-trust environments.

Organisations typically have a team responsible for managing the deployment process. This could be a dedicated DevOps team or a development team who owns the end-to-end lifecycle of a feature, including its deployment to an environment. In all scenarios, the permissions to deploy a configuration change are restricted to a specific user (or group of users). Following this methodology provides organisations with the internal controls and governance required to manage their deployment process.

However this process suffers from multiple flaws; the nature of a dedicated user (or group of users) managing deployments creates a significant dependency on that user. If an urgent patch is required for a production environment and the user who manages deployments is unavailable, it becomes impossible for any changes to be released. By the time the changes have been deployed, the issue could have already caused significant impact.

Autonomous aims to remove the dependency on a centralised group of users in the deployment process. This results in a platform capable of managing software and hardware changes on cloud infrastructure in an autonomous and decentralised manner.

3. Open Source Applications

Although software development can be decentralised via the open source model, open source communities are currently limited to being pseudo-decentralised systems. Foundations (which are centralised) are often required to help support the organisational, legal and financial requirements of open source products. These open source communities lose a significant amount of control due to their dependency on foundations to manage their funding and infrastructure.

In an organisation, the control of funds is equivalent to controlling time and resources. As the allocation of funds directly impacts the prioritisation of work, it could be said that the management of finances is the most important aspect of any system. Foundations act on behalf of their communities when allocating resources (from donations or enterprise sponsorship) but ultimately, their centralised nature means that the true will of decentralised communities may never be exercised - foundation contributors have minimal influence on how their donations are used. Instead of relying on donations managed by foundations, decentralised organisations built on Colony can produce products which generate funds and redirect them into their own colony without an intermediate body (e.g. a foundation or bank). Tasks provide the mechanism for allocating funds across colonies, allowing these communities to prioritise and dedicate resources to the work they deem the most important.

The same principles which describe the importance of controlling funds are closely related to the management of infrastructure. Relying on foundations to support the hardware which runs open source software removes an important aspect of control from decentralised communities; investment into infrastructure and changes to hardware can only be made if approved by the foundation. **Autonomous allows infrastructure to be funded, configured and maintained in a decentralised manner, creating collective ownership of hardware and software in open source communities.**

Colony and Autonomous enables open source communities to manage funds and infrastructure without a centralised body, empowering them to become truly decentralised systems. This will usher in a new era of decentralised technology-centric organisations who can effectively manage themselves without any form of centralised body.

In order to understand how powerful this concept is, imagine an extreme example where a network of crypto-revenue-generating products, developed by decentralised autonomous organisations (DAOs) have their colonies funded by the profits of their product. In this

environment, individuals can be paid by working for these colonies and are given the autonomy to decide what work they perform and for whom.

If software and hardware ownership, production, delivery, funding and maintenance can be decentralised, economic incentives begin to govern the participation of contributors to a product. The DAOs themselves become subject to market forces; well funded and popular applications will encourage strong participation and may progress rapidly, whereas unfunded and unpopular applications will encourage less participation but still enable open-source development. This creates a self-perpetuating cycle of activity and improvements to the ecosystem of the DAOs: higher revenue and rewards result in more development; more development creates a better product; a better product drives higher demand; higher demand leads to more investments; and more investments result in higher revenue and rewards. This is only possible by allowing software and hardware to be managed in a decentralised manner by the DAO itself.

4. Other Applications

Alongside the applications of Autonomous in open source communities, removing the centralised nature of managing infrastructure and the deployment process provides centralised enterprises with multiple benefits.

4.1. Enterprise

Using the Autonomous platform, organisations are not bound by the availability of permissioned users or any factors which are tied to the human-nature of those users (e.g. working hours, location, illness, etc.) meaning that deployments can occur at any time.

Although Autonomous already provides control via the evaluator and dispute process in Colony, multiple layers of criteria could be added to the process such that control remains dynamic and decentralised. For example, by using smart contracts which act as proxy evaluators, many control schemes can be created such as: requiring a certain number of tokens to be staked by members of the colony before a release is finalized; or requiring a quorum of users meeting a reputation threshold to approve a release. This can all be managed by the Autonomous platform.

A practical example of the reputation threshold can be seen in an Enterprise organisation. Instead of depending on permissioned users, a deployment could occur if it has been approved by one or more users with a combined reputation greater than 100 (or any other criteria). This allows multiple combinations of users to approve the deployment which adds flexibility whilst maintaining the appropriate controls (e.g. the deployment could be approved by the Chief Technology Officer, 2 members of the DevOps team or 10 developers).

4.2. Crowdfunded Infrastructure

Companies can create great products adored by their users but fail to reach a sustainable business model. When these businesses cannot generate profit or raise additional funds to continue operating, they are forced to close their company to the dismay of their users and community. Examples of these products include: Vine, a short-form video hosting service; Yik Yak, an anonymous messaging app; and RethinkDB, a scalable database built for real-time applications (which became open source).

Autonomous provides an opportunity for communities to contribute and fund the infrastructure of these services, meaning that popular web-services no longer need to be shutdown due to lack of funds. Users simply create and fund a task via Autonomous to keep their service running on cloud infrastructure. As long as the community has sufficient funds, the service could run indefinitely in its current state with opt-in development and funding driven by the colony's community.

5. Why use Colony?

Colony is a DAO platform that enables flexibility in how funds and actions are allocated and actioned.

This technical proof of concept has little control around who can perform a release. However, extensions can be made to the Autonomous platform via smart contracts to add additional control schemes on top of the evaluation process.

By leveraging reputation score within the colony, staking mechanisms or multi-party verification in contracts that act as "evaluators" in the Colony ecosystem, Autonomous can expand to accomodate a significant amount of functionality around effective infrastructure control.

Colony's funding mechanism also enables cloud infrastructure to be paid for in a decentralised fashion. Beyond the technical demo, Autonomous may accept payments by controlling the smart contract that acts as the evaluator. By claiming the evaluation fee, Autonomous could maintain a pool of funds on behalf of the colony, acting as a intermediary and settling cloud infrastructure payments from this pool.

With the flexibility of Colony's DAO, and the power of the bootstrapped Autonomous platform, a myriad of infrastructure use cases in both enterprise and open source communities can be solved.

6. Technical Implementation

6.1. Architecture

The hackathon entry for Autonomous consists of two components:

- Autonomous decentralised application (DApp)
- Autonomous relayer

These components depend on the following tools and services:

- Kubernetes, specifically a local cluster running Docker for Windows/Mac and Kubernetes
    - With an alternative configuration, Kubernetes can run on Google's Kubernetes Engine (GKE), Amazon's Elastic Container Service for Kubernetes (Amazon EKS) or Azure's Kubernetes Services (AKS)
- Colony, as a decentralised control platform
- Ethereum, as a smart contract platform
- InterPlanetary File System (IPFS), as a distributed file storage system
- Contract Service (similar to TrufflePig but Windows friendly)

By choosing Kubernetes as the engine for controlling infrastructure, we propose that the majority of use cases for modern software and hardware infrastructure can be satisfied by a single-format configuration file.

The current implementation of Autonomous only supports deployments and services, but can theoretically mirror the capabilities of a Kubernetes cloud provider.

6.2. Autonomous Decentralised Application (DApp)

The DApp is the primary interface to the Autonomous platform; it enables actions against the Autonomous relayer as well as colonies that wish to own hardware and software. The hackathon entry provides a simple interface on top of Colony tasks which are tailored to the Autonomous use case. The goal of the Autonomous platform is to allow users to use any Kanban-like Colony interface, enabling them to manage their infrastructure control alongside the other concerns of their colony by using an Autonomous-specific domain. For the technical demo, all tasks which are finalized in the colony are assumed to contain a deliverableHash which contains a Kubernetes-compatible JSON configuration file.

6.3. Autonomous Relayer

The relayer is a service which connects to the colony contracts of registered colonies and listens for events which represent deployments of configurations. In the current implementation, there is no filtering and the relayer will action all FinalizedTasks, however it can be imagined that a colony can organise their configuration tasks by using domains and skills.

Once a task has been completed, the relayer reads the delivery specification from IPFS which contains a JSON Kubernetes configuration, and deploys it to the configured cluster. In addition, the relayer listens to events emitted by these deployments and surfaces them to the user through the DApp. Each colony has its own Kubernetes namespace in the current implementation but more robust schemes of federation can be devised with more research and development.

In order for the relayer to bootstrap itself, it must externalise the configuration of which colonies are registered and which deployments are being listened to. This is performed by using the custom resources API of Kubernetes and having the cluster itself maintain this history and surface it to the relayer on launch. During bootstrapping or during Autonomous infrastructure updates, the relayer and DApp services can be restarted/upgraded within the cluster, and their configuration is still sourceable.

6.4. Hackathon Entry and Demo Capabilities

The capabilities demonstrated in the Autonomous hackathon demo are as follows:

6.4.1. Tasks

- Creating a task with a brief and assigning a worker and evaluator address.
    - For demo purposes, valid addresses are required but they do not impact operation.
- Submit a configuration to complete a task - this submits work to the colony task.
- Finalizing a task - marking the task as approved and completed so that it can be executed by the relayer.

6.4.2. Relayer

- Registering a colony - this allows a colony to be tracked by Autonomous.
- Demo tools to clean the local Kubernetes cluster of existing or residual deployments.
- Viewing deployment events for changes to Kubernetes configuration.

The Autonomous DApp was written to demonstrate the end-to-end lifecycle of a deployment task for the technical demo.

6.5. Bootstrapping

The technical achievement of this demo is successfully demonstrating a relayer which is hosted in a cluster, controlled by a decentralised colony that can instantiate its own upgrade. This feat is enabled through specific configuration of the services used in this demo, and the techniques of configuration externalisation in the cluster as mentioned above.

The bootstrapping demonstration describes how a colony can be created which owns the Autonomous platform to service all other colonies in an open and decentralised fashion. The Autonomous platform can theoretically outlive its creators by leaning on Colony as a DAO.

6.6. Tooling

A few custom scripts and tooling were necessary in the creation of this demo, including: Docker, a container platform provider; a Contract Service similar to TrufflePig; and a command-line interface (CLI) to aid in setting up the demo environment.

7. Current Limitations and Future Improvements

Autonomous has been developed on the assumption that Kubernetes is the system used for automated deployments.

There are no elaborate control mechanisms between Autonomous and other colonies to authorise deployments. As mentioned in previous sections, deployments follow a simple Colony task workflow whereby tasks are assigned to a single evaluator who approves or rejects the deployment. Ideally, deployment tasks would support multi-signature approval via smart contracts, allowing for multiple evaluators to review the changes before a deployment is complete. Multi-signature and smart contract limitations can be overcome through additional development.

In its current state, Autonomous acts as a pseudo-decentralised platform; the services built on the platform are tightly coupled, making it difficult to replace services with minimal effort. This is a minor limitation and Autonomous could be migrated to a more decentralised model in the future.

8. Conclusion

The Autonomous platform provides the ability to deploy software and hardware changes on cloud infrastructure in an autonomous and decentralised manner, including the ability to control its own deployment. Multiple applications of Autonomous have been identified for both centralised and decentralised organisations. Although the use cases are not exhaustive, the

benefits of decentralised autonomous deployment have been described and provide significant advantages compared to a centralised deployment process with permissioned users. The current limitations of Autonomous have been outlined and can be overcome through further development.