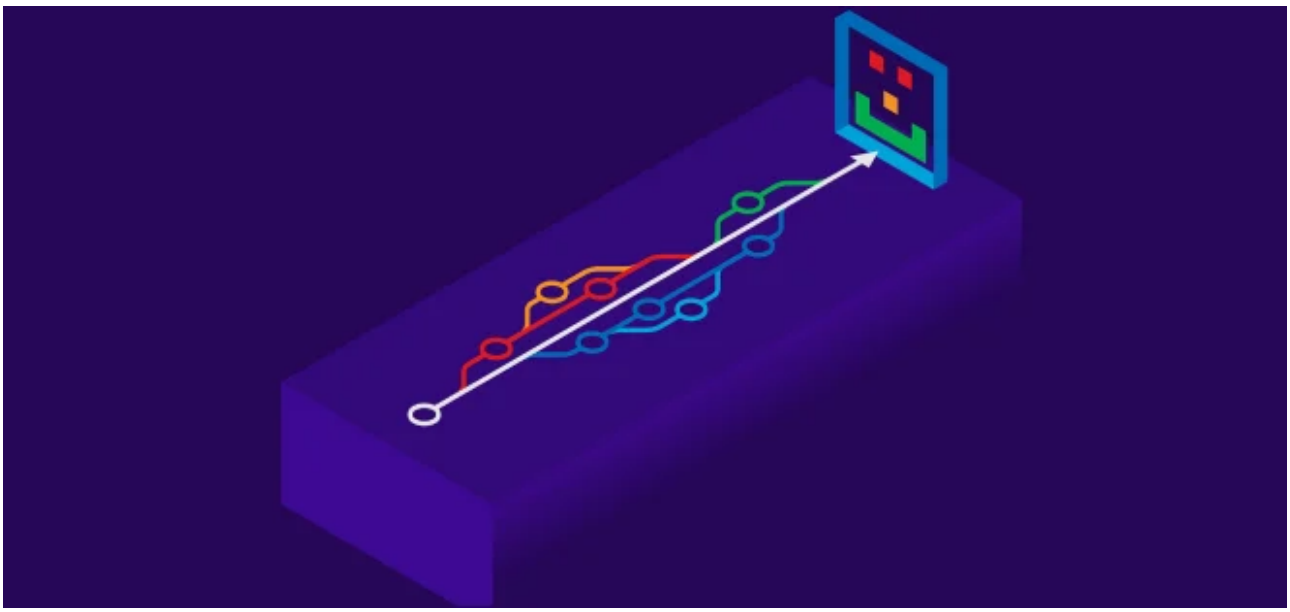
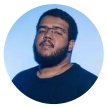
Artigos > **Programação**

uma breve introdução





Emerson Laranja
29/11/2022

Esse artigo faz parte da
Formação Aprenda a programar em JavaScript com foco no back-end

Confira neste artigo:

Introdução

Imagine que você participou do desenvolvimento de um projeto há dois anos no qual utilizou várias bibliotecas e frameworks.

E um problema inesperado aparece: os códigos começam a quebrar e você não consegue seguir devido aos bloqueios de versões das bibliotecas desatualizadas, que dependem de outras bibliotecas. Com isso, surge uma verdadeira “bola de neve”.

Chamamos esse problema de “dependency hell”, ou em português, “inferno de dependências”.

Seria um pesadelo? Quase isso.

Bloqueio de versão é algo que pode acontecer com frequência em projetos grandes que usam muitos pacotes.

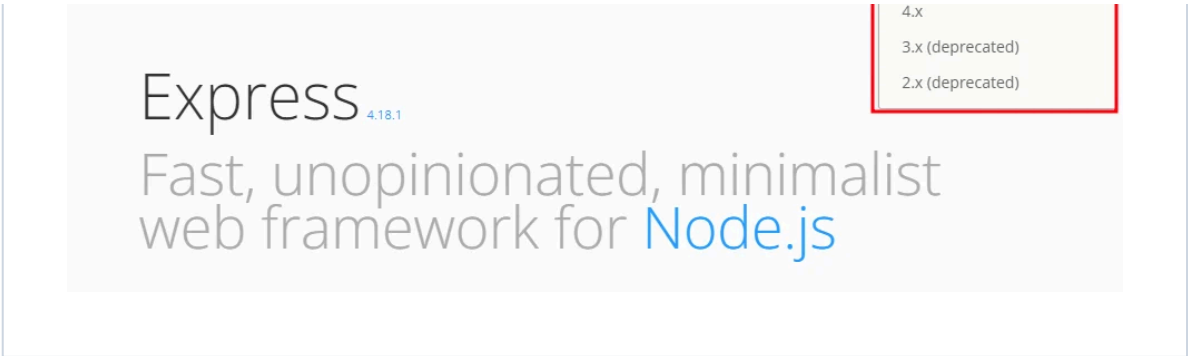
Para evitar o transtorno de ter que sempre ler o último pull request de uma biblioteca, pesquisando features e bugs resolvidos de últimas atualizações, foi criado um padrão que facilita a nomenclatura e comunicação de atualização de versões, o versionamento semântico.

recurso Dependabot.

Definição geral de Versionamento Semântico (SemVer)

> *Versionamento Semântico, em inglês "Semantic Versioning", é um padrão de regras para manter um acompanhamento de versões no desenvolvimento de códigos. Um modelo geral que todas as pessoas usuárias podem entender e utilizar.*

Se você já lida com instalações de pacotes, frameworks e softwares de forma geral, já observou que existe uma sequência numérica que se refere à última versão atualizada.



Na imagem acima, temos o framework Express.js e, na aba *API reference* temos suas diferentes versões.

Cada versão possui um número diferente que, ao clicar em cada uma das versões, vemos melhorias e demais mudanças que podem diferir bastante uma das outras, maior número representa a versão mais atual - no caso 5.x (beta). O *x* indica que podemos ter qualquer número ali, o que importa é o primeiro número da sequência; já o beta está ligado ao momento de lançamento desse software (não se preocupe, veremos isso no próximo tópico).

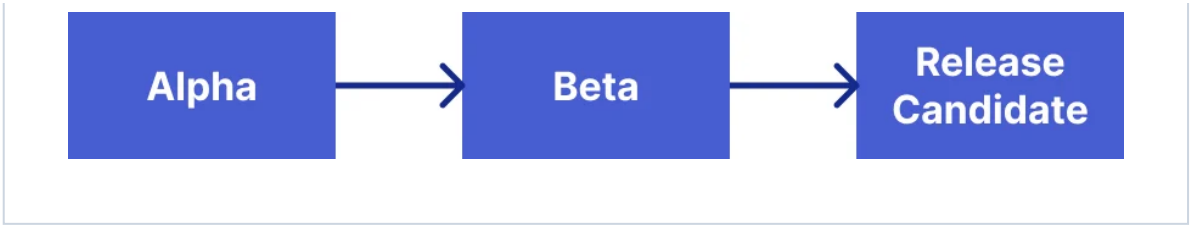
Aqui na Alura procuramos destacar em nossas videoaulas a versão utilizada nas instalações para evitar qualquer tipo de problemas com incompatibilidade de versões e seu código não quebrar no desenvolvimento do projeto.

Esse cuidado deve fazer parte da rotina de uma pessoa desenvolvedora. Versionar seu código é uma forma de acompanhar etapas e comunicar mudanças no projeto, já que constantemente estamos compartilhando códigos através de repositórios públicos e até mesmo privados.

Ciclo de vida de lançamento de um software

Você provavelmente já se deparou com algum anúncio de determinado produto ou jogo que lançou uma versão **beta** para os usuários testarem.

E além da versão beta, nos deparamos com outros nomes ligados ao ciclo de vida de lançamento de um software, que, de forma resumida, estão ligados a determinados momentos (desde seus primeiros dias de vida até o seu lançamento) em que o software se encontra.



- **Alpha**: Além de ser a primeira letra do alfabeto grego, para nosso contexto, essa versão é a primeira a ser testada **pelos times de desenvolvimento** . Essa versão normalmente não é publicada, ou seja, enquanto usuários e usuárias nós ainda não temos acesso às funcionalidades do software, apenas o time interno que fará uma série de testes visando principalmente a estabilidade.
- **Beta**: A segunda letra do alfabeto e que corresponde à versão posterior à alpha. A versão beta se inicia quando as funcionalidades previstas para o software foram implementadas. Essa é a primeira versão pública, onde usuários e usuárias possuem o papel fundamental de testar o software, assim a empresa saberá como está a usabilidade de seu produto e como reduzir impactos negativos relacionados a ele.

É muito comum que *beta testers* (users que testam a versão beta) recebam recompensas, como descontos ou até receber o software gratuitamente (no caso de produtos pagos).

- **RC (Release Candidate)**: Uma tradução seria “candidato a lançamento”, ou seja, trata-se da versão com potencial de ser o produto final, aquele que foi testado em uma versão beta, se mostrou estável, suas funcionalidades estão bem especificadas e o software está pronto para ser lançado (a menos que algum bug mais severo se perceba a tempo).

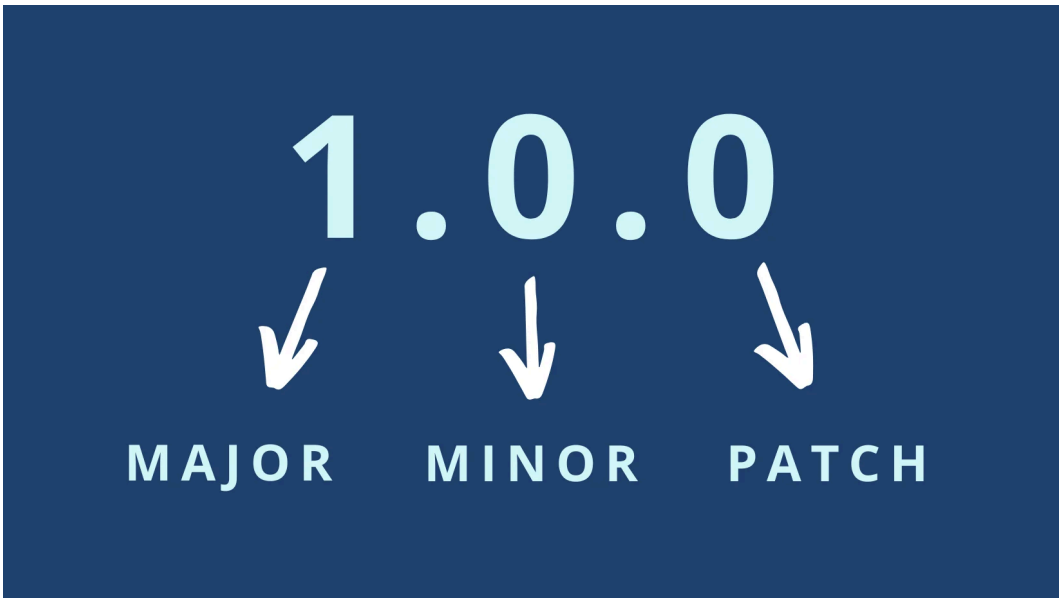
Bom! Até aqui já entendemos como funciona o lançamento de um software. Mas assim como o exemplo do Express, após seu lançamento, o software recebe desde pequenas correções como mudanças grandes no seu comportamento.E, para cada mudança, uma nova versão é lançada.

Para termos um versionamento semântico, não faz sentido que um software mude da versão 1 para versão 2 tanto para mudanças pequenas quanto grandes. Isso acaba não sendo significativo para nós, pois você não

E, pensando nisso, existe um formato para uma versão: basta apenas olhar sua estrutura e podemos identificar que tipo e o impacto de uma mudança foi feita de uma versão para outra.

Formato do Versionamento Semântico

O padrão do versionamento é uma sequência numérica separada em três posições:



MAJOR (Maior)

O primeiro dígito informa a versão de compatibilidade e é alterado caso o software ou biblioteca sofra mudanças que a torne incompatível com outras versões. São as chamadas *breaking changes*, atualizações que possuem o potencial de "quebrar" códigos que utilizam versões anteriores.

Exemplo: você está usando uma função de uma biblioteca X, porém foi lançada uma nova versão da biblioteca onde essa função tem outro nome ou

MINOR (Menor)

O segundo dígito informa a versão da funcionalidade, onde uma nova função ou melhoria substancial é adicionada e não há problemas de incompatibilidade com outras versões.

Exemplo: A biblioteca que você costuma usar tem agora uma nova funcionalidade e é compatível com outras versões, necessita apenas de atualização local.

Versão 2.0.0 → Agora é 2.1.0

PATCH (Correção)

O terceiro dígito informa a versão da correção de bugs, melhorias de desempenho ou alterações similares que não alteram as funcionalidades atuais e nem introduzem novas.

Exemplo: A biblioteca que você costuma usar tem um bug que gera uma vulnerabilidade no código. Esse bug foi corrigido em uma nova versão.

Versão 2.1.0 → Agora é 2.1.1

Regras de publicação do software

O versionamento semântico traz algumas especificações visando manter uma comunicação clara entre os softwares, para que você, ao olhar o formato de uma versão, tenha o mesmo entendimento que qualquer outra pessoa ao redor do mundo. Algumas regras são:

- Um número de versão tem o formato X.Y.Z, onde X, Y e Z são inteiros não negativos e não devem conter zeros à esquerda. X é a versão major (maior, em português), Y é a versão minor (menor, em português) e Z é a versão patch (correção, em português).

- Quando um pacote for lançado (released), o conteúdo dessa versão não deve ser modificado! Se houver modificações, uma nova versão deve ser lançada.
- Uma versão de pré-lançamento (como vimos anteriormente: alpha, beta, rc) pode ser identificada quando adicionado um hífen após a versão de Correção (o Z do nosso formato) e uma série de identificadores separados por ponto. O identificador deve incluir apenas caracteres alfanuméricos e hífen [0-9A-Za-z-] e não deve ser vazio. Exemplos: 1.0.0-alpha, 1.0.0-alpha.1, 1.0.0-0.3.7.
- No início do desenvolvimento, a versão Maior deve ser zero (0.y.z).
- A versão 1.0.0 define a API como pública (aquela que foi lançada e está disponível para o público). A maneira como o número de versão é incrementado após este lançamento é dependente da API pública e como ela muda.

Caso queira entender outras regras acerca do versionamento semântico, você pode consultar a documentação oficial clicando aqui.

Controlando versões no Git/Github

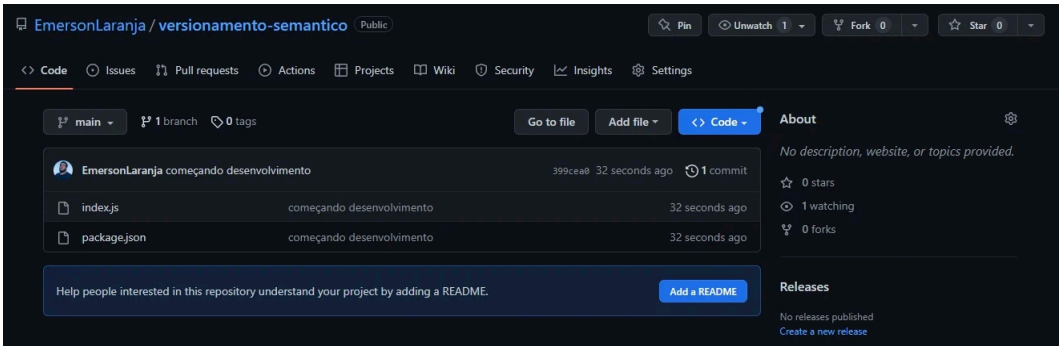
Sabemos que o Git nos permite fazer o controle das versões do nosso código (caso esse assunto seja novo para você, recomendo fortemente que leia esse artigo sobre Git e Github para entender do que estou falando aqui).

Porém, além dos commits que indicam as mudanças feitas no nosso código, podemos unir um conjunto de funcionalidades através das tags e releases (em inglês).

Basicamente, as tags representam os trechos de código de uma versão e através dos releases nós podemos descrever para o público do que se trata essa versão.

Assim, você pode indicar nos seus projetos (tanto para seu time quanto para o público externo) qual a versão atual do projeto na totalidade, tanto para quando estiver começando o desenvolvimento, quanto seus códigos

versionamento-semantico, fiz o clone no editor que estou usando (no caso, Visual Studio Code) e já realizei o commit de um trecho de código (a inicialização de um servidor em Node.js). Através do Github, esse é o nosso ponto de partida:



> *Novamente, caso sinta dificuldade com algum desses passos, recomendo, além da leitura do artigo, o curso Git e Github: Controle de versão para por em prática esses conceitos.*

E para criarmos nossa primeira tag, em nosso terminal aberto no projeto, digitamos o seguinte comando:

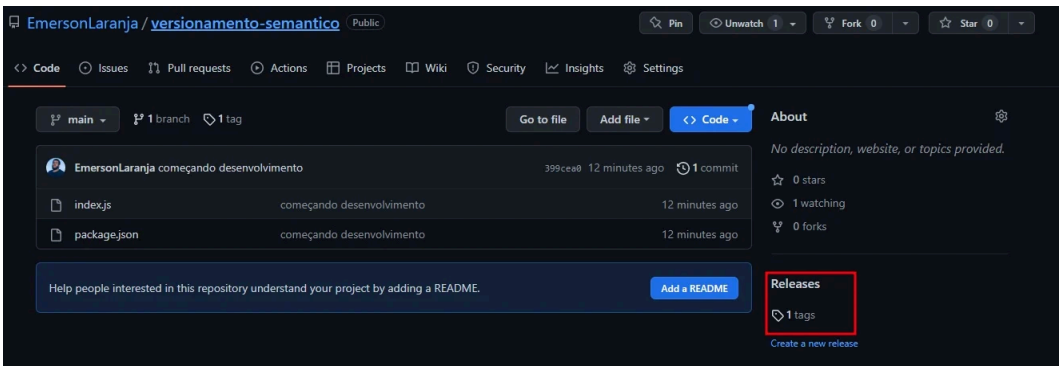
```
git tag -a <NUMERO_DA_TAG> -m “<MENSAGEM_DA_TAG>”
```

Assim, para indicar que estamos começando o desenvolvimento do projeto, criando nossa primeira versão (0.1.0, por exemplo), podemos digitar:

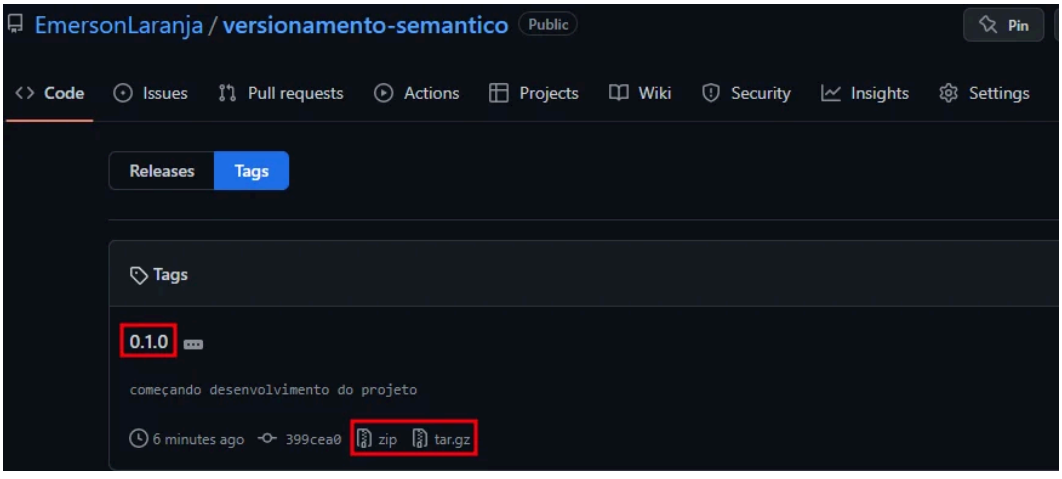
```
git tag -a 0.1.0 -m “começando o desenvolvimento do projeto”
```

E para enviar as tags para nosso repositório remoto, digitamos o comando:

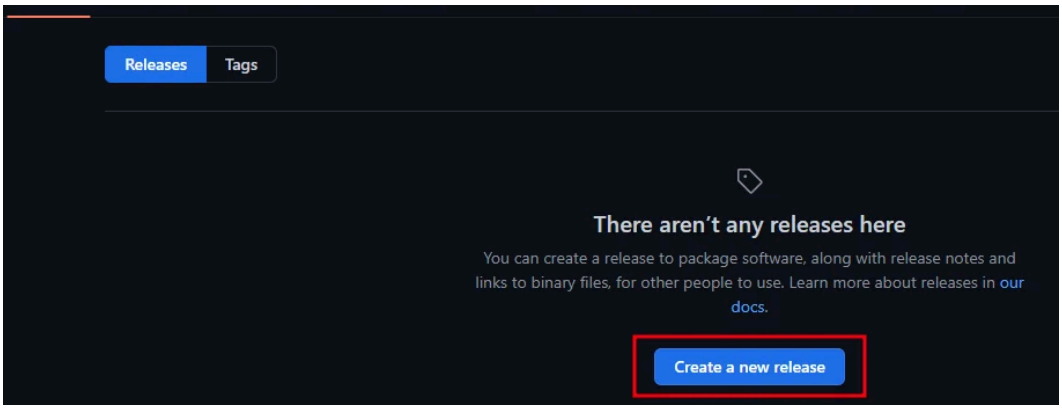
Agora, ao atualizar a página do nosso projeto no Github, vemos que o campo *Releases* recebeu uma alteração, conforme é mostrado a seguir:



Ao clicar em “1 tags” vemos a tag que criamos, bem como a mensagem de descrição e a possibilidade de fazer o download desta versão, conforme a sequência abaixo:



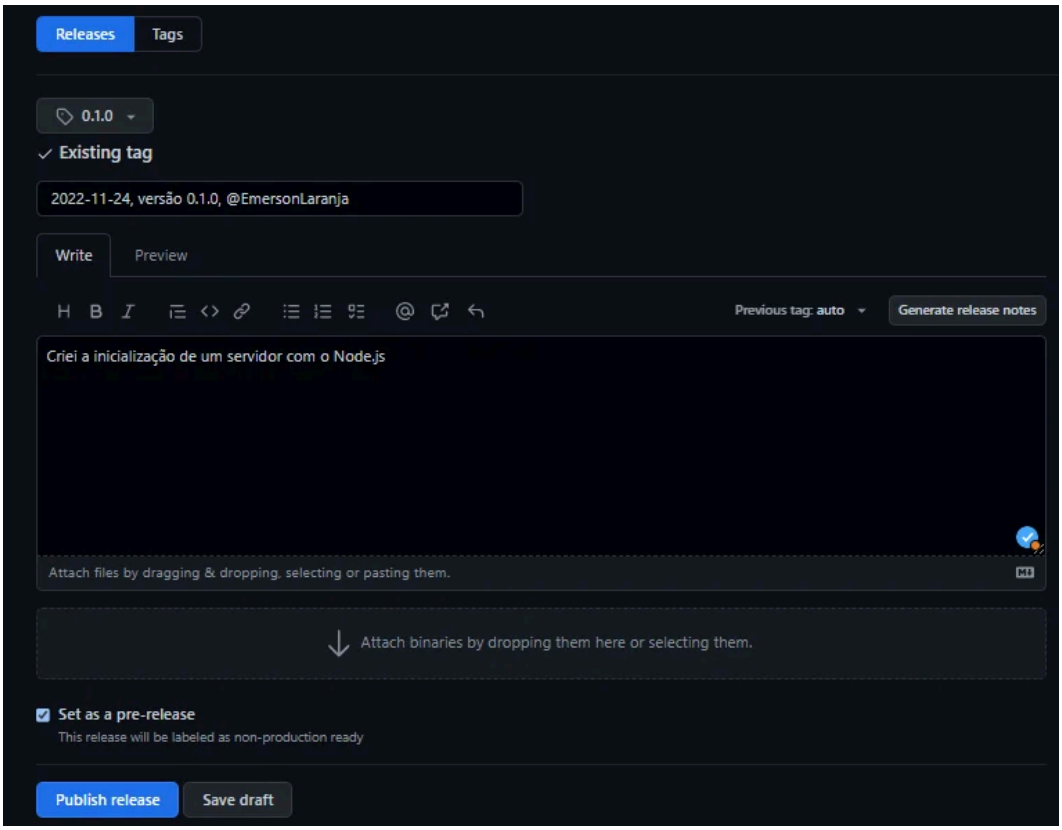
Por fim, ao acessar a aba “Releases”, podemos criar uma versão de lançamento, para que as pessoas consigam visualizar do que se trata essa primeira versão, clicando em “Create a new release”. Confira este passo na tela a seguir:

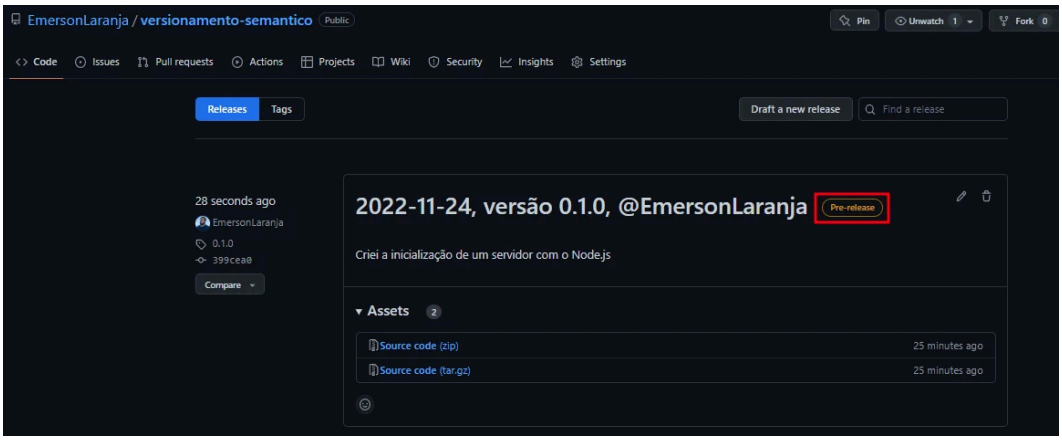


Após isso, basta selecionar a tag que se refere essa release (no caso, 0.1.0), preencher com um título (é comum usar uma estrutura que indique a data e a versão da release, adicionei também o usuário responsável) e uma descrição.

Você pode também selecionar a opção "Set as a pre-release" caso essa ainda não seja uma versão de lançamento (no caso, eu optei por marcar , pois é apenas o começo do nosso desenvolvimento do projeto).

Confira na tela a seguir:





Pronto! Temos nossa primeira release publicada e qualquer pessoa que acessar nosso projeto saberá como está o andamento do projeto (no caso, “prerelease”, como indicado na imagem acima), bem como as funcionalidades desta versão.

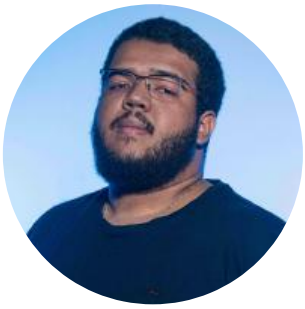
Um exemplo mais elaborado de detalhes é a release da última versão do Node.js, na qual vemos a descrição das novas funcionalidades, commits relacionados e o código referente a esta versão.

Conclusão

Neste artigo vimos o que é o versionamento semântico, como identificar de forma rápida o seu formato (graças às regras de padronização das versões), além de usá-lo de forma prática, por exemplo, nos seus projetos do Git.

Esse conhecimento será essencial para tomar decisões de quando atualizar seus projetos, bem como versioná-lo. Mas para que esses conceitos se tornem um aprendizado, além de entender mais detalhes na documentação oficial, é necessário que você pratique!

Aqui na Alura você pode praticar com diversos projetos, como sua primeira biblioteca usando Node.js, onde você também poderá lançar sua biblioteca no npm, praticando mais do versionamento semântico.



Emerson Laranja

Sou monitor da Alura e granduando em engenharia de computação (Ufes).Minha dedicação está centrada no desenvolvimento de conteúdos voltados para a área de backend, com enfoque especial em JavaScript e TypeScript. Estou comprometido em proporcionar uma experiência de aprendizado envolvente e enriquecedora para todos os alunos, contribuindo assim para o sucesso de suas jornadas no universo do desenvolvimento web.

Leia também:

- Node.JS: o que é, como funciona esse ambiente de execução JavaScript e um Guia para iniciar
- Guia de JavaScript: o que é e como aprender a linguagem mais popular do mundo?
- Java: o que é, linguagem e um Guia para iniciar na tecnologia

Veja outros artigos sobre
Programação

MATRICULE-SE

Nossas redes e apps

Institucional

Sobre nós

Trabalhe conosco

Para Empresas

Para Sua Escola

Política de Privacidade

Compromisso de Integridade

Termos de Uso

Status

A Alura

Formações

Como Funciona

Todos os cursos

Depoimentos

Instrutores(as)

Dev em <T>

Luri, a inteligência artificial da Alura

Conteúdos

Alura Cases

Imersões

Artigos

Podcasts

Artigos de educação corporativa

Fale Conosco

Email e telefone

Perguntas frequentes

Email* ENVIAR

CURSOS

Cursos de Programação
Lógica | Python | PHP | Java | .NET | Node JS | C | Computação | Jogos | IoT

Cursos de Front-end
HTML, CSS | React | Angular | JavaScript | jQuery

Cursos de Data Science
Ciência de dados | BI | SQL e Banco de Dados | Excel | Machine Learning | NoSQL | Estatística

Cursos de Inteligência Artificial
IA para Programação | IA para Dados

Cursos de DevOps
AWS | Azure | Docker | Segurança | IaC | Linux

Cursos de UX & Design
Usabilidade e UX | Vídeo e Motion | 3D

Cursos de Mobile
React Native | Flutter | iOS e Swift | Android, Kotlin | Jogos

Cursos de Inovação & Gestão
Métodos Ágeis | Softskills | Liderança e Gestão | Startups | Vendas